ULTRIX

digital

Reference Pages
Section 8: Maintenance

Order Number: AA–LY20B–TE

ULTRIX

Reference Pages Section 8: Maintenance

Order Number: AA-LY20B-TE
June 1990

Product Version: ULTRIX Version 4.0 or higher

This manual describes commands for system operation and maintenance for system administrators on both RISC and VAX platforms.

digital equipment corporation
maynard, massachusetts

About Reference Pages

The *ULTRIX Reference Pages* describe commands, system calls, routines, file formats, and special files for RISC and VAX platforms.

Sections

The reference pages are divided into eight sections according to topic. Within each section, the reference pages are organized alphabetically by title, except Section 3, which is divided into subsections. Each section and most subsections have an introductory reference page called `intro` that describes the organization and anything unique to that section.

Some reference pages carry a one- to three-letter suffix after the section number, for example, `scan`(1mh). The suffix indicates that there is a ''family'' of reference pages for that utility or feature. The Section 3 subsections all use suffixes and other sections may also have suffixes.

Following are the sections that make up the *ULTRIX Reference Pages*.

Section 1: Commands

This section describes commands that are available to all ULTRIX users. Section 1 is split between two binders. The first binder contains reference pages for titles that fall between A and L. The second binder contains reference pages for titles that fall between M and Z.

Section 2: System Calls

This section defines system calls (entries into the ULTRIX kernel) that are used by all programmers. The introduction to Section 2, `intro`(2), lists error numbers with brief descriptions of their meanings. The introduction also defines many of the terms used in this section.

Section 3: Routines

This section describes the routines available in ULTRIX libraries. Routines are sometimes referred to as subroutines or functions.

Section 4: Special Files

This section describes special files, related device driver functions, databases, and network support.

Section 5: File Formats

This section describes the format of system files and how the files are used. The files described include assembler and link editor output, system accounting, and file system formats.

Section 6: Games

The reference pages in this section describe the games that are available in the unsupported software subset. The reference pages for games are in the document *Reference Pages for Unsupported Software.*

Section 7: Macro Packages and Conventions

This section contains miscellaneous information, including ASCII character codes, mail addressing formats, text formatting macros, and a description of the root file system.

Section 8: Maintenance

This section describes commands for system operation and maintenance.

Platform Labels

The *ULTRIX Reference Pages* contain entries for both RISC and VAX platforms. Pages that have no platform label beside the title apply to both platforms. Reference pages that apply only to RISC platforms have a ''RISC'' label beside the title and the VAX-only reference pages that apply only to VAX platforms are likewise labeled with ''VAX.'' If each platform has the same command, system call, routine, file format, or special file, but functions differently on the different platforms, both reference pages are included, with the RISC page first.

Reference Page Format

Each reference page follows the same general format. Common to all reference pages is a title consisting of the name of a command or a descriptive title, followed by a section number; for example, `date`(1). This title is used throughout the documentation set.

The headings in each reference page provide specific information. The standard headings are:

Name	Provides the name of the entry and gives a short description.
Syntax	Describes the command syntax or the routine definition. Section 5 reference pages do not use the Syntax heading.
Description	Provides a detailed description of the entry's features, usage, and syntax variations.
Options	Describes the command-line options.
Restrictions	Describes limitations or restrictions on the use of a command or routine.
Examples	Provides examples of how a command or routine is used.

Return Values	Describes the values returned by a system call or routine. Used in Sections 2 and 3 only.
Diagnostics	Describes diagnostic and error messages that can appear.
Files	Lists related files that are either a part of the command or used during execution.
Environment	Describes the operation of the system call or routine when compiled in the POSIX and SYSTEM V environments. If the environment has no effect on the operation, this heading is not used. Used in Sections 2 and 3 only.
See Also	Lists related reference pages and documents in the ULTRIX documentation set.

Conventions

The following documentation conventions are used in the reference pages.

%	The default user prompt is your system name followed by a right angle bracket. In this manual, a percent sign (%) is used to represent this prompt.
#	A number sign is the default superuser prompt.
`user input`	This bold typeface is used in interactive examples to indicate typed user input.
`system output`	This typeface is used in text to indicate the exact name of a command, routine, partition, pathname, directory, or file. This typeface is also used in interactive examples to indicate system output and in code examples and other screen displays.
UPPERCASE lowercase	The ULTRIX system differentiates between lowercase and uppercase characters. Literal strings that appear in text, examples, syntax descriptions, and function definitions must be typed exactly as shown.
rlogin	This typeface is used for command names in the Syntax portion of the reference page to indicate that the command is entered exactly as shown. Options for commands are shown in bold wherever they appear.
filename	In examples, syntax descriptions, and routine definitions, italics are used to indicate variable values. In text, italics are used to give references to other documents.
[ ]	In syntax descriptions and routine definitions, brackets indicate items that are optional.
{ | }	In syntax descriptions and routine definitions, braces enclose lists from which one item must be chosen. Vertical bars are used to separate items.

. . .	In syntax descriptions and routine definitions, a horizontal ellipsis indicates that the preceding item can be repeated one or more times.
. . .	A vertical ellipsis indicates that a portion of an example that would normally be present is not shown.
`cat`(1)	Cross-references to the *ULTRIX Reference Pages* include the appropriate section number in parentheses. For example, a reference to `cat`(1) indicates that you can find the material on the `cat` command in Section 1 of the reference pages.

Online Reference Pages

The ULTRIX reference pages are available online if installed by your system administrator. The `man` command is used to display the reference pages as follows:

To display the `ls`(1) reference page:

```
% man ls
```

To display the `passwd`(1) reference page:

```
% man passwd
```

To display the `passwd`(5) reference page:

```
% man 5 passwd
```

To display the Name lines of all reference pages that contain the word ''passwd'':

```
% man -k passwd
```

To display the introductory reference page for the family of 3xti reference pages:

```
% man 3xti intro
```

Users on ULTRIX workstations can display the reference pages using the unsupported `xman` utility if installed. See the `xman`(1X) reference page for details.

Reference Pages for Unsupported Software

The reference pages for the optionally installed, unsupported ULTRIX software are in the document *Reference Pages for Unsupported Software*.

Name

intro – introduction to system maintenance and operation commands

Description

This section contains information related to system operation and maintenance. In particular, commands used to create new file systems, `newfs`, `mkfs`, and verify the integrity of the file systems, `fsck`, `icheck`, `dcheck`, and `ncheck` are described here. The `format`(8v) reference page should be consulted when formatting disk packs. The `crash`(8v) reference page should be consulted in understanding how to interpret system crash dumps.

2780d (8)

Name

2780d – IBM 2780/3780 RJE emulator daemon

Syntax

2780d

Description

The `2780d` program is automatically executed when a user runs `2780e`(1) or `3780e`(1). Once executed, `2780d` transmits all data files (df*) in `/usr/spool/rje` which have associated control files (cf*), to the IBM system.

Two users executing `2780e (3780e)` do not cause two daemons to run. The first one executed handles both transmissions. The daemon's process id is kept in `/usr/spool/rje/.rjed`. Each time a daemon is run, it checks that file to see if an active daemon is running and exits if one is.

If `2780d` has trouble accessing files, the system manager should clean up the spool directory by removing any outdated received files or data files which are not associated with a corresponding control file.

The `2780d` program may become disabled when: the call to the IBM system has not been completed within one minute, the IBM system has gone down during transmission, the `/usr/spool/rje/.rjed` file can not be read, there has been an error in sending or receiving.

To re-establish the connection to the IBM system, restart the daemon by typing `2780d` on the command line.

When files are received from the IBM system, `2780d` places them in `/usr/spool/rje/rjetemp.out`. If your username is included in the job control statements, the files are given a file name with the form *rjedaytime* and placed in your home directory. If your username is not included, the files are placed in `/usr/spool/rje`. If there are problems in renaming the files, they remain in `/usr/spool/rje/rjetemp.out`.

To obtain a log of all files that are sent and received the system manager must create `/usr/spool/rje/acctlog`. Once created, this file will list each file transmission. Errors are logged in `/usr/adm/rjelog`.

Files

`/usr/spool/rje/.rjed`	Process identification file
`/usr/spool/rje/cf*`	Control file
`/usr/spool/rje/df*`	Data file
`/usr/spool/rje/zf*`	Renamed control file
`/usr/spool/rje/acctlog`	File transmission log file

`/usr/adm/rjelog` Error log file

See Also

2780e(1), 3780e(1)

Name

ac – login accounting

Syntax

/etc/ac [**–w** *wtmp*] [**–p**] [**–d**] [*people*] ...

Description

The `ac` command produces a printout giving connect time for each user who has logged in during the life of the current *wtmp* file. A total is also produced.

The accounting file `/usr/adm/wtmp` is maintained by `init` and `login`. Neither of these programs creates the file, so if it does not exist no connect-time accounting is done. To start accounting, it should be created with length 0. On the other hand if the file is left undisturbed it will grow without bound, so periodically any information desired should be collected and the file truncated.

Options

–w	Specifies an alternate *wtmp* file.
–p	Prints individual totals; without this option, only totals are printed.
–d	Causes a printout for each midnight to midnight period. Any *people* will limit the printout to only the specified login names. If no *wtmp* file is given, `/usr/adm/wtmp` is used.

Files

`/usr/adm/wtmp`

See Also

last(1), lastcomm(1), login(1), utmp(5), init(8), sa(8)

Name

addnode – add or change an entry in the nodes database

Syntax

/etc/**addnode** *node* [*options*]

Description

The `addnode` command adds a new node entry to the nodes database or modifies an existing entry. The nodes database is the one used by DECnet. The `addnode` command defines the information that is necessary for your ULTRIX node to be capable of down-line loading and up-line dumping a particular target node. The node address is the address of the target node. If the target node is a DECnet node, then the node address is mandatory and the node name is optional. However, if the target node is a non-DECnet node, for example, a terminal server, you should specify only the node name and omit the node address.

If you do not specify an absolute pathname for secondary load, tertiary load, system load, diagnostic load, or dump file, the default path, `/usr/lib/mop,` is used during the load or dump process.

A node address is a decimal integer in the range of 1 to 1023 for single area networks, or has the format *a.n* for multiarea networks, where *a* is the network area number (a decimal integer in the range of 1 to 63) and *n* is the node number (a decimal integer in the range of 1 to 1023).

The node name is the node name of the target node.

If the target node is a DECnet node, the node name is optional. However, if the target node is a non-DECnet node, for example, a terminal server, the node name is mandatory (and the node address should be ignored).

A node name can be from 1 to 6 alphanumeric characters, including at least 1 alphabetic character.

Options

–A Uses the specified host address (next argument) as that of the target node.

–D Creates an up-line memory dump of the target node in the specified file (next argument).

–N Uses the specified name (next argument) as that of the target node.

–c Uses the specified service and device number (UBA-n or QNA-n) as the circuit to the target node.

–d Sends the specified diagnostic load image (next argument) to the target node.

–h Uses the specified address (next argument) as the Ethernet address of the target node.

–l Sends the specified system load image (next argument) to the target node.

–p Uses the specified service and password (next arguments) in accessing the target node.

–s Sends the specified secondary load file (next argument) to the target node.

–t Sends the specified tertiary load file (next argument) to the target node.

Examples

```
% /etc/addnode mynode -h aa-00-03-00-01-19 \ <RET>
-s /usr/download/secondary \ <RET>
-t /usr/download/tertiary \ <RET>
-l system <RET>
```

This command adds the non-DECnet, node mynode, to the nodes database, which has the Ethernet physical address aa-00-03-00-01-19. This command also specifies the file names for the secondary loader, the tertiary loader, and the system loader. Note that a path name is not specified for the system loader; consequently, the loader uses the default path `/usr/lib/mop` when searching for that file.

```
# /etc/addnode 44.71 -h aa-00-03-00-01-20 <RET>
```

This command adds the DECnet node 44.71 to the nodes database, which has the Ethernet physical address aa-00-03-00-01-20.

See Also

ccr(8), getnode(8), load(8), mop_mom(8), remnode(8), trigger(8)

Name

adduser, removeuser, addgroup – add and remove user accounts

Syntax

/etc/adduser
/etc/removeuser
/etc/addgroup

Description

The `adduser` command provides an interactive facility for adding new user accounts to the system password file.

The `adduser` command also sets up a home directory with the files `.cshrc`, `.login`, and `.profile` from the `/usr/skel` directory for new users.

The interactive `removeuser` command deletes user accounts from the system password file. It also gives the option of deleting the user's home directory and files.

The interactive `addgroup` command adds new groups to the `/etc/group` file.

Restrictions

The 'name' can contain only lower case ASCII characters a to z and the numbers 0 to 9.

Files

`/etc/utmp`	Lock file
`/etc/passwd`	Password file
`/etc/group`	Group file
`/usr/skel`	Default files directory

See Also

passwd(1), passwd(5yp), vipw(8)

ansi_ps (8)

Name

ansi_ps, regis_ps, tek_ps – datatype to PostScript (TM) translators

Syntax

ansi_ps [*options*]

regis_ps [*options*]

tek4014_ps [*options*]

Description

The `ansi_ps` command is the ANSI to PostScript (TM) translator.

The ANSI translator implements DEC-STD-074-0 *(Printer System Reference Manual)* and conforms to ISO/DSI 6429, ISO/DSI 2022, and ANSI X3.4 standards.

Escape sequences (documented in the translator reference manual) not beginning with DEC are from ISO/DSI 6429, ISO/DSI 2022 and ANSI X3.4 standards. Escape sequences beginning with DEC are legal extensions of DEC-STD-074-0.

The ANSI translator does not implement many escape sequences documented in the above standards.

The `regis_ps` command is the Regis to PostScript (TM) translator.

The `tek4014_ps` command is the Tektronix 4014 to PostScript (TM) translator.

Each translator reads from standard input and writes to standard output. If the data type is to be printed on a PostScript (TM) printer with specialised support, the translator is invoked by `lpd` using the `xlator_call` script. Refer to `xlator_call(8)`.

Options

The valid options for the `ansi_ps` translator are:

–e Send <CSI>20h, setting linefeed/newline.

–F*pagesize*
Selects the size of the pages to be printed. The valid page sizes are: *letter, a, ledger, b, legal, executive, a5, a4, a3, b5* or *b4*. If the page size is not specified, *a4* is used.

–O*orientation*
Selects the orientation of the text on the page. The valid orientations are: *portrait* or *landscape*. If the orientation is not specified, *portrait* is used.

–m*output_mode*
Selects the ouput mode of `ansi_ps`. The output modes are:

8f	Full 8-bit output
7f	7-bit output
8g	GL/GR only
7g	GL only

If the output mode is not specified, *8f* is used.

–R*resource_string*
Informs `ansi_ps` of a pre-loaded resource present in the PostScript (TM) environment of the printer. Multiple **–R** options may be passed.

–s Inhibits the final showpage. This allows more than one page to be printed on each sheet.

The valid options for the `regis_ps` and `tek4014_ps` translators are:

–F*pagesize*
Selects the size of the pages to be printed. The valid page sizes are: *letter, a, ledger, b, legal, executive, a5, a4, a3, b5* or *b4*. If the page size is not specified, *a4* is used.

–O*orientation*
Selects the orientation of the text on the page. The valid orientations are: *portrait* or *landscape.* If the orientation is not specified, *portrait* is used.

–s Inhibits the final showpage. This allows more than one page to be printed on each sheet.

See Also

ln03rof(8), lpd(8), xlator_call(8)

ap(8mh)

Name

ap – parse addresses RFC 822-style

Syntax

/usr/new/lib/mh/ap [–form *formatfile*] **[–format** *string*] **[–normalize] [–nonormalize] [–width** *columns*] **addrs ... [–help]**

Description

The ap program parses addresses according to the ARPA Internet standard. It also understands many non-standard formats. It is useful for seeing how MH will interpret an address.

The ap program treats each argument as one or more addresses, and prints those addresses in the official RFC 822-format. Hence, it is usually best to enclose each argument in quotation marks (" ") for the shell.

Options

To override the output format used by ap, you use the -format *string* or -format *file* options. This permits individual fields of the address to be extracted easily. The *string* argument is a format string; the *file* argument is the name of a format file. See mh-format(5mh) for the details.

In addition to the standard escapes, scan also recognizes the following additional escape: error, which is a diagnostic that is returned if the parse failed.

If the -normalize option is given, ap tries to track down the official hostname of the address.

Here is the default format string used by ap:

```
%<{error}%{error}: %{text}%|%(proper{text})%>
```

The previous example says if an error was detected, print the error, a colon (:), and the address in error. Otherwise, output the RFC 822-proper format of the address.

The argument to the -format option must be interpreted as a single token by the shell that invokes ap. Therefore, you should usually place the argument to this option inside quotation marks (" ").

Defaults

-format	Defaults as described previously.
-normalize	
-width	Defaults to the width of the terminal.

Restrictions

On systems where MH was configured with the BERK option, address parsing is not enabled. The version of MH that is supplied with ULTRIX Mail Connection has the BERK option enabled.

Files

`$HOME/.mh_profile`
User profile

`/usr/new/lib/mh/mtstailor`
Tailor file

See Also

mh-format(5mh), dp(8mh), *Standard for the Format of ARPA Internet Text Messages* (RFC 822)

arcv (8)

Name

arcv – convert archives to new format

Syntax

/etc/arcv *file ...*

Description

The `arcv` command converts archive files from 32v and Third Berkeley editions to a new portable format. For further information, see `ar`(1) and `ar`(5). The conversion is done in place, and the command refuses to alter a file not in old archive format.

Old archives are marked with a magic number of 0177545 at the start. New archives have ''!<arch>'' as their first line.

Files

`/tmp/v*`	Temporary copy

See Also

ar(1), ar(5)

Name

arff – archiver for RT-11 format devices

Syntax

/etc/**arff** *key*

Description

The `arff` command manipulates RT-11 formatted devices, such as the console media on VAX computers, or RT-11 formatted device images stored as a file on your ULTRIX system. The *key* argument is a single letter, possibly followed by one or more letters. Possible values for *key* follow:

Keys

d Name files to be deleted from the RT-11 device. Note that protected files cannot be deleted. Wild cards are available with this function. At least one RT-11 file name or wild card specification must be given.

i The RT-11 device is initialized. The RT-11 device home block and directory segments are rewritten. This effectively deletes all files on the RT-11 device.

Exactly one argument must be given instead of a file name. This argument is a comma-separated list of parameters to use in initializing the device:

1. The first parameter is the device size in blocks (8 to 65535); this parameter must be given.
2. The second parameter is the directory size in segments (1 to 31); this parameter may be omitted.
3. The third parameter is the volume identification (1 to 12 characters); this parameter may be omitted.
4. The fourth parameter is the volume owner (1 to 12 characters); this parameter may be omitted.
5. The fifth parameter is the number of extra words per directory entry (0 to 500); this parameter may be omitted.

Parameters may be omitted by not placing a value between the separating commas. Trailing commas need not be provided.

The default value for the number of directory segments depends on the specified device size. For sizes up to 640 blocks, 1 directory segment is assumed; for 641 to 1280 blocks, 2 segments; 1281 to 2560 blocks, 4 segments; 2561 to 5120 blocks, 8 segments; 5121 to 10240 blocks, 16 segments; and 10241 or more blocks, 31 segments. This approximates, but does not duplicate, the RT-11 defaults.

The default value for the volume identification is RT11A; for the volume owner, blanks; and for the number of extra words per directory segment, 0. The ''system identification'' in the home block is set to ''DEC ULTRIX''; this cannot be changed by command options.

p The named files are protected against deletion. Wild cards are available with this function. If no RT-11 file names or wild card specifications are given, all files on the RT-11 device are protected against deletion.

r The *key-arguments* name files to be replaced on the RT-11 device. If the named files do not exist on the RT-11 device, they are added to the RT-11 device. Wild cards are not available with this function.

t The *key-arguments* name files to be listed. Wild cards are available with this function. If no RT-11 file names or wild card specifications are given, all files on the RT-11 device are listed.

u The *key-arguments* name files to be made unprotected against deletion. Wild cards are available with this function. If no RT-11 file names or wild card specifications are given, all files on the RT-11 device are protected against deletion.

x The *key-arguments* name files to be extracted from the RT-11 device. Wild cards are available with this function. If no RT-11 file names or wild card specifications are given, all files on the RT-11 device are extracted into the current working directory.

b A boot block is written to an RT-11 device. Currently this works on VAX 8600 media only. The *key-arguments* name the monitor program file and the device handler program file, in that order. The default files if no *key-arguments* are specified are rt11fb.sys and dl.sys. The monitor program file and the device handler file are first extracted from the RT-11 device and a boot block is constructed. The default or named files will not be extracted if the ''n'' (no extract) modifier is given.

Key Modifiers

The *key-modifier* letters and their effects are:

c Create new directory. This modifier indicates that a new (one-segment) directory is to be created on the RT-11 device. This modifier may be specified only with the **r** (replace) function. The home block is not modified, and, if corrupt, will remain corrupt.

f File name. This modifier indicates that the first file name specified in the command is the name of a native file which contains an RT-11 device image. This file is used rather than the default of /dev/bootdev (the real device name should be linked to this). This modifier may be specified with any function.

h Home block corrupt. This modifier indicates that the home block of the RT-11 volume is known to be corrupt. Directory information contained in the home block is ignored in favor of usually valid assumptions. This modifier may be necessary when dealing with RT-11 format volumes created by the arff program supplied with 4.2bsd and ULTRIX Version 1.0.

Note that if the home block is noticed to be corrupt and this modifier is not given, a warning message is issued and this modifier is assumed.

m No sector mapping. This modifier indicates that the device driver mapping is to be used. If this modifier is not given on ULTRIX systems, the RX01 mapping used by the Digital proprietary operating systems is applied to the RT-11 device image. This modifier should not be specified when the RT-11 device image is the VAX-11/780 or VAX-11/785 RX01 diskette or a physical image thereof, and should be specified in all other cases. On operating system other than ULTRIX, this modifier is assumed and its specification has no effect. This modifier may be specified with any function.

p Printable files. This modifier indicates that the files transferred to and from the RT-11 device are printable text files. When extracted from the RT-11 device, NUL characters are deleted and CR LF sequences are replaced by newline. When replaced into the RT-11 device, newline is replaced by CR LF sequences. When this modifier is not specified, trailing NUL characters are deleted on extraction, and sufficient NUL characters to fill out a block are appended on replacement. This modifier may be specified only with the **r** (replace) and **x** (extract) functions.

v Verbose. When specified with the **t** (list) function, the listing produced is that which would be produced by the RT-11 command

```
DIRECTORY /FULL /POSITION /VOLUME /COLUMNS:2
```

rather than the RT-11 command

```
DIRECTORY /BRIEF /COLUMNS:6
```

When specified with the **i** (initialize) function, the volume parameters specified and implied by the defaulting rules are written to the standard output. When specified with the **r** (replace), **x** (extract), **d** (delete), **p** (protect), or **u** (unprotect) functions, the names of the files affected are listed on the standard output as they are processed.

n No extract. Do not extract the monitor program and the handler code from the RT-11 device prior to writing a boot block. Used only on VAX 8600 RT-11 device and with the ''b'' option given above.

File Names

Most of the functions take a list of file names specifying the files upon which the functions are to be performed. RT-11 file names may be specified in three formats:

RT-11 file names
These consist of one to six characters followed by a period followed by zero to three characters. The characters must be in the set A-Z, 0-9, and dollar sign. The characters before the period are known as the *filename*; the characters before the period are known as the *extension*.

RT-11 wild card specifications
These consist of zero to six characters optionally followed by a period optionally followed by one to three characters. The characters must be in the set A-Z, 0-9, dollar sign, percent sign, and asterisk. Percent sign will match any single character in the RT-11 file names on the RT-11 volume; asterisk will match zero or more characters in the RT-11 file names on the RT-11 volume. An RT-11 wild card specification may be distinguished from an RT-11 file name by the lack of a filename, the lack of a period and extension, or the use of at least one * or % wild card character. A null filename is equivalent to *. No string is equivalent to *.*.

Native file names
These consist of one or more characters optionally followed by a period optionally followed by one or more characters. If no period appears, the file

name must be at least seven characters long. Native file names generate RT-11 file names as follows: If there is no period or if there are at least seven characters before the first period, the filename is the first six characters and the extension is up to the next three characters, terminated by the end of the native file name or the appearance of a period. If there are one to six characters before the first period, the filename is those characters and the extension is up to the next three characters, terminated by the end of the native file name or the appearance of another period.

In any format, path information may be prepended.

When native file names are generated (for the extract function), the form of the generated file name depends upon the file name argument specified. All prepended path information is present verbatim. If the argument is an RT-11 wild card specification, the last component of the full path name is the lowercase equivalent of the RT-11 file name. Otherwise, the last component of the full path name is exactly that specified in the argument, with uppercase and lowercase intact. Thus, `/usr/include/` as an argument to the extract function would extract all files on the RT-11 device (a null specification is equivalent to the *.* wild card), and store them in the directory `/usr/include` with file names the lowercase equivalent of the RT-11 file names; but `/usr/include/Makefile` as an argument to the extract function would extract the RT-11 file MAKEFI.LE and store it as `/usr/include/Makefile`.

Examples

The following example produces a full listing of the files on the RT-11 device image in the file ''device_image''.

```
arff tvf device_image
```

The following example produces a short listing of the files with extension ''C'' on the RT-11 device image in the file ''device_image''.

```
arff tf device_image .c
```

The following example extracts the file FILENAME.C from the RT-11 device at the standard place and places it in the file `/usr/users/kmd/FILENAME.C` (with uppercase).

```
arff x /usr/users/kmd/FILENAME.C
```

The following example protects all files on the RT-11 device at the standard place against deletion.

```
arff p
```

The following example unprotects all files on the RT-11 device at the standard place with extension ''OBJ'' against deletion, and notes each file so unprotected on the standard output.

```
arff uv .obj
```

The following example causes an error message.

```
arff d
```

The following example initializes the file test_device to be an RT-11 device image with two directory segments, 494 total blocks, and one extra word per directory entry.

```
arff ivf test_device 494,2,,,1
```

The following example extracts the files rt11fb.sys and dl.sys from the RT-11 device, constructs a boot block, and writes the boot block back to the device.

```
arff bm
```

The following example uses the files rt11a.sys and dd.sys to contruct a boot block for an RT-11 device. This boot block is then written to the device.

```
arff bnm rt11a.sys dd.sys
```

Files

`/tmp/arff*`	RT-11 file image for the ''replace'' function

Name

arp – address resolution display and control

Syntax

arp –a [*vmunix*] [*kmem*]
arp [**–d**] *hostname*
arp –f *filename*
arp –s *hostname ether_addr* [*temp*] [*pub*] [*trail*]

Description

The `arp` program displays and modifies the Internet-to-Ethernet address translation tables used by the address resolution protocol `arp`(4p).

The *hostname* is the name of the host system for which information will be displayed or modified.

With no flags, the program displays the current ARP entry for *hostname*.

Options

–a Displays current Address Resolution Protocol (ARP) entries from the specified name list and core files (next arguments). If not specified, uses `/vmunix` and `/dev/mem`, respectively.

–d Deletes the entry for the host specified by name (next argument).

–f Reads information from the specified file (next argument) and modifies entries accordingly. Entries in the file should be of the folowing form, with argument meanings as given previously:

```
hostname ether_addr [ temp ] [ pub ] [ trail ]
```

–s Creates an ARP entry for the host called *hostname* with the Ethernet address *ether_addr*. The Ethernet address is given as six hexadecimal bytes separated by colons. The entry will be permanent unless the word *temp* is given in the command. If the word *pub* is given, the entry will be published. That is, the system will act as an ARP server, responding to requests for *hostname* even though the host address is not its own. The word *trail* indicates that trailer encapsulations may be sent to this host.

See Also

arp(4p), ifconfig(8c)

Name

audgen – generate an audit record

Syntax

audgen *audit_record_parameter_list*

Description

The `audgen` utility generates an audit record, which gets placed in the auditlog. Standard audit event information, such as identification information and timestamp, are automatically inserted. The *audit_record_parameter_list* consists of 1 to 8 strings, each of which gets inserted into the audit record. The event type is set to AUDGEN8.

Restrictions

This utility makes use of the `audgen`(2) system call, which requires privilege. No record will be generated if the event AUDGEN8 is not being audited for the current process. The maximum number of arguments referenced by argv is AUD_NPARAM (8).

See Also

audgen(2)

Name

auditd – audit daemon

Syntax

/etc/sec/auditd [*options* ...]

Description

The audit daemon, `auditd`, operates as a server, monitoring `/dev/audit` for local audit data, monitoring a known port for data from remote cooperating audit daemons, and monitoring an AF_UNIX socket for input from the system administrator.

Local audit data is read from the `/dev/audit` device. Data read from `/dev/audit` is buffered by the audit daemon, and eventually output into the auditlog when the buffer nears capacity or the daemon receives an explicit instruction from the administrator to flush its buffer.

Local administrative data is read via the socket `/tmp/.audit/audS`. Input from the system administrator allows for changing of the daemon's configurable options. The administrator communicates with the audit daemon by executing `auditd` with the desired options. The first invocation of `auditd` spawns the daemon; subsequent invocations detect that an audit daemon already exists and will communicate with it, passing along directions for the selected options. The first invocation of the daemon also turns on auditing for the system (`audcntl`(2)). When the daemon is terminated, by the `-k` option or the SIGTERM signal, auditing is turned off. It is important not to have system auditing turned on when there is no audit daemon running on the system (processes being audited will sleep until `/dev/audit` is read, which is typically done by the audit daemon).

Remote audit data is first detected when the remote audit daemon attempts to communicate with the local audit daemon. To establish a communications path between the remote and the local daemons, the remote audit daemons hostname is first checked against a list of hosts allowed to transmit data to the local host. This list is maintained in `/etc/auditd_clients.` If the remote host is allowed to transfer audit data to the local host, a child audit daemon dedicated to communicating with the remote host is spawned.

Options

-a Toggle the KERBEROS switch. If on, KERBEROS authentication routines will be used to verify the identity of any audit daemons attempting to communicate. This occurs either when sending to a remote host (by the `-i` option) or accepting from remote hosts (by the `-s` option).

-b *alternate_pathname*
Sets the pathname to which the audit daemon will write its data should the location currently accepting data become unavailable. This can happen should the current location specify a remote host which is no longer available, or when the filesystem of the current location reaches an overflow condition (in this case, the alternate pathname must specify a partition other than the currently overflowing partition).

-c *pathname* Sets the pathname to which the audit daemon will post any warning or informational messages (such as "audit log change"). This may be either a device or local file.

-d Causes the audit daemon to dump its currently buffered audit data out to `/dev/audit`. The audit daemon normally dumps its buffer only when it approaches capacity.

-f *percentage* Sets the minimum percent free space on the current partition before an overflow condition is triggered.

-h Outputs a brief help menu.

-i *hostname* Causes the audit daemon to transfer its audit data to the audit daemon executing on the remote host *hostname*. If the remote site stops receiving, the local daemon will store its data locally (in *alternate_pathname* if available).

-k Kills the audit daemon (killing the local daemon turns audit off).

-l *pathname* Causes the audit daemon to output its audit data to the local file *pathname*.

-n *kbytes* Sets the size of the audit daemons buffer for the audit data (minimum is 4).

-o *overflow action*
Sets the system action to take on a local overflow condition. Alternatives are a) use the alternate log specified via -b option, b) shutdown the system, c) switch to the root-mounted filesystem with the most free space, d) suspend auditing until space is made available, and e) overwrite the current auditlog.

-p *daemon id* Specifies the id of the audit daemon to receive the current options. When the local audit daemon accepts a connection to receive data from a remote audit daemon, a dedicated child audit daemon is spawned off from the local audit daemon to service that connection. With this scenario, multiple audit daemons may exist on a single system. Specifying the id of the `auditd` allows for communication with one of the child audit daemons. The id for each daemon can be found by entering the following at the command line:

```
/etc/sec/auditd -?
```

The previous command line displays the current options. No id's are displayed unless at least one child audit daemon exists. If the `-p` option is not specified when running with more than one audit daemon, the master daemon (accepting audit data for the local system) handles the request. When the master daemon is killed, it kills all of its child daemons.

-q Queries the audit daemon for the current location of the audit data.

-s Toggles the network server switch. If on, allows the audit daemon to accept audit data from other audit daemons whose hostnames are specified in the `/etc/auditd_clients` file.

-t *timeout value*
Sets the timeout value used in establishing initial connections with remote audit daemons.

-x Auditlog pathnames are always appended with a suffix consisting of a generation number. These generation numbers range from 0 to 999. (Generation numbers may be overridden via explicit generation number specification on the pathname for the `-lfR option, for example auditlog.345).` The `-x` option causes a change in auditlog to the next auditlog in the generation number sequence. (If the current log was auditlog.345, then `-x` would change the log to auditlog.346). Whenever an auditlog is closed, it is also compressed (by `/usr/ucb/compress`).

-z Removes any AF_UNIX sockets left by previous daemons. This occurs when the system shuts down abnormally. This option is useful typically only for the `auditd` invocation from the `/etc/rc.local` file. If no AF_UNIX socket is present, the next invocation of `auditd` will start the `audit` daemon. If an AF_UNIX socket is present, the next invocation of `auditd` will spawn a client process which will communicate with the system audit daemon. This `-z` option removes any leftover AF_UNIX sockets, forcing a new audit daemon to start. This should be used only when no audit daemon is present on the system.

-? Shows the current status of the audit daemons options.

Files

`/etc/auditd_clients`

See Also

audcntl(2), audit(4)

Name

auditmask – get or set system-call event and trusted-event audit masks

Syntax

auditmask [*event*[*:succeed:fail*] | **-f[ull]** | **-n[one]**]

Description

The `auditmask` command with no arguments displays the system-calls and trusted-events currently being audited for the system, and displays whether they are being audited under successful or failed occurrences or both. The format used for the display is acceptable as input to the `auditmask` command.

The `auditmask` command with arguments sets the system-call and trusted-event audit masks for the system. This is cumulative operation, so it is possible to turn on or off audit for one set of events, then turn on or off audit for a second set of events without changing the first set of events (except for intersection between the two sets). Command line arguments to `auditmask` can include one or more events, each with an optional field *:succeed:fail*, where *succeed* is either 0 to specify no auditing of successful occurrences of *event*, or 1 to specify auditing of successful occurrences of *event*; and *fail* is either 0 to specify no auditing of failed occurrences of *event* or 1 to specify auditing of failed occurrences of *event*. The event name is either the system-call name or the trusted-event name (see `audit.h`).

The `auditmask` command will also accept redirected input, which can be the output of a previously issued `auditmask` command. This is a file which contains lines of the format *event*[*succeed*] [*fail*]. If *succeed* is present, successful occurrences of that event will be audited; if *fail* is present, failed occurrences of that event will be audited; if both are present, successful and failed occurrences will be audited; if neither is present, that event will not be audited.

The `auditmask` command is used in `/etc/rc.local` to initialize the audit mask at boot time according to the file `/etc/sec/audit_events`. This makes use of privileged operations within the `audcntl`(2) system call.

Options

–f	Turns on full auditing for the system. This list may include events for X or events which are represented by a number (reserved for future use); these events will not be audited, despite their presence in the auditmask.
–n	Turns off all auditing for the system.

See Also

audcntl(2)

Name

audit_tool – ULTRIX auditlog reduction tool

Syntax

/usr/etc/sec/audit_tool [*option* ...] *auditlog_filename*

Description

The `audit_tool` presents a human-understandable format of selected portions of the collected audit data. If no arguments are provided, a brief help message will be displayed. The auditlog file may be compressed or uncompressed. The `audit_tool` command will uncompress the auditlog file if necessary, and re-compress it if it was originally compressed.

Options are used to select specific audit records of interest. For a record to be selected, it must match at least one option of each option type specified. For example, if two usernames and one hostname were specified, an audit record to be selected would have to match one of the usernames and the hostname. Only one start/end time may be selected. Only one deselection rulesfile may be selected. It is possible to select as many events as exists on the system. For all other option types, up to 8 instances may be selected.

Options

-a *audit_id* Selects audit records with a matching *audit_id*. The default is to select for all *audit_id*'s.

-b Outputs selected records in binary format. The output is in a format suitable for analysis by the `audit_tool`. The default is to output in ASCII format.

-B Outputs selected records in an abbreviated format. Each selected event is displayed along with its audit_id, ruid, result, error code, pid, event name, and parameter list. Suppressed information includes the username, ppid, device id, current directory, gnode information, symbolic name referenced by any descriptors, IP address, and timestamp. The default is to output in the non-abbreviated format.

-d *filename* Reads deselection rules from the specified file and suppress any records matching any of the deselection rules. The deselection rulesets take precedence over other selection options. Each deselection rule is a tuple consisting of hostname, audit_id, ruid, event, pathname, and flag. The flag component is used to specify read or write mode; it pertains only to open events. Wildcarding and simple pattern matching are supported. Take, for example, the following lines from a deselection file:

```
# HOST, AUID, RUID, EVENT, PATHNAME, FLAG
* * * open /usr/lib/* r
grumpy * * * /usr/spool/rwho* *
```

These lines indicate that any open operations for read access on any object whose pathname starts with `/usr/lib/` will not be selected, and on system **grumpy** any operations performed on any object whose pathname starts on `/usr/spool/rwho` will not be selected. (Lines beginning with number signs (#) are treated as comment lines). Any field can be replaced with an asterisk (*), which indicates a match with any value. Pathname matching requires an exact match between strings, unless the pathname is suffixed with an asterisk, which matches any string (so, for example, `/usr/spool/rwho*` matches `/usr/spool/rwho/anything`). The default is to apply no deselection rulesets. (Specifying the `-D` option instead of `-d` will additionally print the deselection rulesets to be applied).

-e *event[:success:fail]*
Selects records with a matching event. Optionally select only those records with a successful/failed return value. For example, the option `-e open:0:1` selects for only failed open events. Multiple events may be specified on the command line. The default is to select for all events, both successful and failed.

-E *error* Selects records with a matching error. The default is to select for all errors.

-f Causes the `audit_tool` not to quit at and end-of-file, but to continue attempting to read data. This is useful for reviewing auditlog data as it is being written by the audit daemon. (For SMP systems, audit data should be sorted first, as descriptor translation, loginname, current directory, and root directory all rely on state information maintained by the `audit_tool`).

-g *gnode_id* Selects records with a matching gnode identifier number. The default is to select for all gnode id's.

-G *gnode_dev major#,minor#*
Selects records with matching gnode device major/minor numbers. The default is to select for all gnode devices.

-h *hostname/IP address*
Selects records with a matching hostname or IP address. Hostnames are translated to their IP addresses via the local `/etc/hosts` file. If the local `/etc/hosts` is not available or contains insufficient information, IP addresses should be used. The default is to select for all hostnames and IP addresses.

-i Enter interactive selection mode to specify options. Interactive mode may also be entered by hitting CTRL/C at any time, then specifying ''no'' to the exit prompt. Once in interactive mode, each option will be selected for. Press Return to accept the current setting (or default); enter an asterisk (*) to change the current setting back to the default. The default, unless otherwise stated, is to select every audit record.

-o Whenever the audit daemon switches auditlogs, an audit_log_change event is generated. If that event did result in an auditlog change (that is, it was an event which occurred on the local system), the `audit_tool` will normally attempt to find and process the succeeding auditlog. This is possible, however, only if the auditlog is maintained locally. The **-o** option tells the `audit_tool` not to process succeeding auditlogs.

-p *pid* Selects records with a matching pid. The default is to select for all pids.

-P *ppid*	Selects records with a matching parent pid (ppid). The default is to select for all ppids.
-r *ruid*	Selects records with a matching read uid (ruid). The default is to select for all ruids.
-R	Generates an ASCII report for each *audit_id* found in the selected events. Each report consists of those events selected which have an *audit_id* matching that of report suffix. Report names are of the format report.xxxx, where xxxx is the *audit_id.*
-s *string*	Selects records which contain *string* in either a parameter field or a descriptor field. The default is to select for all strings.
-S	Performs a sort (by time) on the auditlog. The sort performed is an inter-cpu sort only (for any specific cpu, data may be non-sequential for events such as fork and vfork; this information does not need to be sorted for proper operation of the reduction tool). This option is useful only for data collected on an SMP system.
-t *start_time*	Selects records which contain a timestamp no earlier than *start_time.* Timestamp format is *yymmdd[hh[mm[ss]]].* The default is to select for all timestamps.
-T *end_time*	Selects records which contain a timestamp no later than *start_time.* Timestamp format is *yymmdd[hh[mm[ss]]].* The default is to select for all timestamps.
-u *uid*	Selects audit records with a matching uid. The default is to select for all uid's.
-U *username*	Selects audit records with a matching username. Usernames are recorded at the *login* event and are associated with all child processes. If *login* is not audited, no username will be present in the auditlog. Selecting for a *username* will display those records which have a matching username. The default is to select for all usernames.
-x *major#,minor#*	Selects audit records with matching device major/minor numbers. The default is to select for all devices.

The audit reduction tool generates auditlog header files, suffixed with .hdr, when it completes processing of a auditlog file. If the **-o** option is used, no auditlog header file is generated. This header file contains the time range in which the audited operations occurred, so searching for events by time requires only those auditlogs which were actually written into during that time to be processed by the reduction tool. The header file also contains the sort status of the auditlog, so previously sorted logs don't get sorted more than once.

Restrictions

The audit reduction tool maintains the state of each process in order to translate descriptors back to pathnames, as well as provide current working directory, root, and username. In order not to run out of memory, `exit`(2) should be an audited event. In order to provide current working directory, `chdir`(2) should be an audited event. In order to provide current root (if not /), `chroot`(2) should be an audited event. In order to provide username, login should be an audited event.

All state relevant information current at the time of an auditlog change is maintained in the header file. This allows subsequent scans of a specific auditlog to not have any dependencies on previous auditlogs.

Examples

The following example selects all **login**, **open** and **creat** events performed on system **grumpy** by any process with audit_id 1123:

```
audit_tool -e login -e open -e creat -h grumpy -a 1123 auditlog.000
```

The following example applies deselection file *deselect* to auditlog.000 and selects for events between 10:47 a.m. on April 13, 1986 and 5:30 p.m. on April 20, 1986:

```
audit_tool -d deselect -t 8604131047 -T 8604201730 auditlog.000
```

See Also

auditd(8), auditmask(8)

Name

automount – automatically and transparently mounts and unmounts NFS file systems

Syntax

/usr/etc/automount [**–mnTv**] [*–D name= value*] [**–f** *master-file*]
[**–M** *mount-directory*] [**–tl** *duration*] [**–tm** *interval*] [**–tw** *interval*]
[*directory mapname* [*– mount-options*]]

Description

The `automount` command transparently mounts and unmounts NFS file systems on an as-needed basis. It is useful for mounting file systems and directories that are needed only occasionally and it provides an alternative to using `/etc/fstab` for NFS mounting file systems on client machines.

The `automount` program can be started from the `/etc/rc.local` file or the command line. The daemon forked by the `automount` program sleeps until a user attempts to access a directory that is associated with an automount map. The daemon then consults the appropriate map and mounts the NFS file system. If the indicated directory has not already been created, the daemon creates it and removes it after automatic unmount. Automount maps are typically located in the `/etc` directory but can be placed in any directory. The maps indicate where to find the file system to be mounted, the local mount point, and mount options. After a specified period of inactivity on a file system, 5 minutes by default, the `automount` daemon unmounts that file system.

An individual automount map is either local or served by the Yellow Pages. A system, however, can use both local and Yellow Pages automount maps. When a map is referenced, the `automount` program first looks for the designated mapname locally. If it cannot find the mapname locally, it looks for a Yellow Pages map by that name. The names of the maps are passed to `automount` from the command line, or from a master map.

If there are contradictory arguments on the command line and in the master map, those on the command line take precedence.

By default, the daemon mounts the remote file system under the directory `/tmp_mnt` and creates a symbolic link between the requested and the actual mount points.

Maps

Conventionally, `automount` maps are files that are located in the `/etc` directory with names that have the prefix `auto.` They indicate what remote file systems to mount, where to mount them, and with what options.

The Master Map

The `automount` program can consult a master map, which contains entries that point to other maps that can be either direct or indirect. If Yellow Pages is running, `automount` checks for the presence of a YP map named `auto.master.` You are not required to run YP or have an `auto.master` map. A master map can also be a file whose location is specified with the `-f` command line option.

The master map provides `automount` a list of maps, and arguments that pertain to each of the maps. The syntax for each line in the master map is:

mount-point map [mount-options]

- *Mount-point* is the full pathname of a local directory if the *map* argument is the name of an indirect map or the name of a special map. If the *map* argument is the name of a direct map, the dummy directory ''/-'' is specified as the *mount-point.*
- *Map* is the name of the map the automount command uses to find the mount points and locations. This can either be a filename, a YP map name, or a special map name.
- *Mount-options* is a list of options used to regulate the mounting of entries listed in *map*.

Direct Maps

Direct maps specify what remote file systems to mount locally and what the local mount points are. They do not point to other maps. They also can specify mount options. The syntax for direct maps is:

key [mount-options] location

- *Key* is the full pathname of the mount point.
- *Mount-options* are the options for this specific mount. They are optional, but if present override mount options specified on the command line or in the master map.
- *Location* is the location of the resource being mounted, specified as: *server:pathname*. Multiple *location* fields can be specified, in which case `automount` sends multiple mount requests and mounts from the first server to respond.

Indirect Maps

Indirect maps have the same format as direct maps. The only difference between a direct and an indirect map is that the key in a direct map is a full pathname, whereas the key in an indirect map is a simple name that does not begin with a slash. (Remember that the indirect map as a whole has been associated with a directory specified in the master map or on the command line. The entries in an indirect map list subdirectories that are individually mounted within the directory associated with the map.)

Special Maps

The `-hosts` map is a special `automount` map that is used to access all directories exported by a server to a client.

The following command allows a client to access directories that are exported from any host in its `/etc/hosts` file or the Yellow Pages hosts database if the client is running YP.

```
# automount /net -hosts
```

For example, suppose that hera and sheba are both hosts on a local area network that is running Yellow Pages. If the /etc/rc.local file on hera contains the command automount /net -hosts, then users on hera can access any directories that sheba exports to hera. All of the exported directories are mounted under /net/sheba on hera.

The -null map, when indicated on the command line, cancels the map associated with the directory indicated. It can be used to cancel a map specified in the master map. For example, invoking the automounter with:

```
# automount /net -null
```

causes the /net entry in auto.master to be ignored.

Pattern Matching

The ampersand (&) is expanded into the key field in a map wherever it appears. For example, in this case, the amptersand (&) expands to oak:

```
#key   mount_options        location
#
oak                         &:/export/&
```

The asterisk (*), when supplied as the key field, is recognized as the catch-all entry. It is used to substitute for lines that are all formatted similarly. The automount program uses the asterisk to match any hostname not listed as a key in an entry before the asterisk. Any entry following the asterisk is ignored. An example of pattern matching in a map follows:

```
#key   mount_options        location
#
oak                         &:/export/&
*                           &:/home/&
```

Environment Variables

The value of an environment variable can be used within an automount map by prefixing a dollar sign ($) to its name. You can also use braces to delimit the name of the variable from appended letters or digits. The environment variables can be inherited from the environment or can be explicitly defined with the -D command line option.

Hierarchical Mounts

You can mount different directories within an automount file system hierarchy from different servers. For example, if you are mounting the /usr/local file system on your machine, you can mount the various subdirectories within /usr/local from different servers.

In the following example, the directories /usr/local, /usr/local/bin, /usr/local/src, and /usr/local/tools are mounted from the machines host1, host2, host3, and host4 respectively. When the root of the hierarchy is referenced, the automount program mounts the whole hierarchy.

```
/usr/local\
             /      -ro,soft      host1:/usr/local \
             /bin   -ro,soft      host2:/usr/local/bin \
             /src   -ro,soft      host2:/usr/local/src \
             /tools -ro,soft      host2:/usr/src/tools
```

Readability has been improved by splitting the entry into five lines and indenting the continuation lines.

Options

–m
: Ignores directory-mapname pairs listed in the `/etc/auto.master` Yellow Pages database.

–n
: Disables dynamic mounts. Lookups intercepted by the `automount` daemon succeed when the target file system has been previously mounted.

–T
: Traces all NFS requests received by the daemon. Information about the details of the request are expanded and sent to standard output.

–v
: Logs status messages to the console. (Stands for ''verbose''.)

–D *name=value*
: Defines an `automount` environment variable by assigning *value* to the variable.

–f *master-file*
: Uses *master-file* for a list of initial directory to mapname pairs, ahead of the `auto.master` Yellow Pages map. If an entry exists in both *master-file* and `auto.master,` that specified in *master-file* is used since it is read first. Similarly, entries on the command line take precedence over *master-file* entries. This technique can be used to replace entries in global maps with your own.

–M *mount-directory*
: Uses *mount-directory* instead of the default, `/tmp_mnt`.

–tl *duration*
: Specifies a *duration* in seconds, that a file system is to remain mounted when not in use. The default is 5 minutes.

–tm *interval*
: Specifies an *interval* in seconds, between attempts to mount a file system. The default is 30 seconds.

–tw *interval*
: Specifies an *interval* in seconds, between attempts to unmount file systems that have exceeded their cached times. The default is 1 minute.

–mount_options
: Specifies the mount options to be applied to all of the directories listed in *mapname*. If mount options are listed in the specified map, they take precedence over these options.

NOTE

Sending the SIGTERM signal to the `automount` daemon causes it to unmount all file systems that it has mounted, and to exit.

Sending the SIGHUP signal to the `automount` daemon causes it to reread the system mount table to update its internal record of currently-mounted file systems. If a file system mounted with `automount` is unmounted by a umount command, automount should be forced to reread the system mount table.

Restrictions

Shell filename expansion does not apply to objects not currently mounted.

Since automount is single-threaded, any request that is delayed by a slow or non-responding NFS server will delay all subsequent automount requests until it completes.

Examples

The following is a sample auto.master map:

```
#
# mount-point       mapname              mount-options
#
/net                -hosts
/home               /etc/auto.indirect   -rw,intr,secure
/-                  /etc/auto.direct     -ro,intr
```

The following is a typical automount indirect map:

```
#
# key         mount-options       location
#
john                              merge:/home/merge/john
mary                              stripe:/home/stripe/mary
fred                              blur:/usr/staff/fred
```

The following is a typical automount direct map:

```
#
# key         mount-options       location
#
/usr/source   -ro                 merge:/usr/src/proto
/usr/local                        blur:/usr/bin/tools
```

The following is a sample indirect map that specifies multiple mount locations for the file system reference. The first server to respond to a mount request gets mounted:

```
reference       -ro,soft        earl:/usr/src/ref\
                                fern:/usr/staff/ron/ref\
                                irv:/usr/backup/reference
```

Files

/tmp_mnt	Directory where automounted file systems reside

See Also

mount(8), mount(8nfs), umount(8)
Guide to the Network File System

Name

bad144 – read/write DEC Standard 144 bad sector information

Syntax

/etc/bad144 [**–f**] disktype disk [sno [bad ...]]

Description

The `bad144` command can be used to inspect the information stored on a disk that is used by the disk drivers to implement bad sector forwarding. The format of the information is specified by DEC Standard 144, as follows.

The bad sector information is located in the first five even numbered sectors of the last track of the disk pack. There are five identical copies of the information, described by the *dkbad* structure.

Replacement sectors are allocated starting with the first sector before the bad sector information and working backwards towards the beginning of the disk. A maximum of 126 bad sectors are supported. The position of the bad sector in the bad sector table determines which replacement sector it corresponds to. The bad sectors must be listed in ascending order.

The bad sector information and replacement sectors are conventionally only accessible through the ''c'' file system partition of the disk. If that partition is used for a file system, the user is responsible for making sure that it does not overlap the bad sector information or any replacement sectors.

The bad sector structure is as follows:

```
struct dkbad {
long         bt_csn;         /* cartridge serial number */
 u_short     bt_mbz;         /* unused; should be 0 */
 u_short     bt_flag;        /* -1 => alignment cartridge */
 struct bt_bad {
        u_short bt_cyl;      /* cylinder number of bad sector */
        u_short bt_trksec;   /* track and sector number */
 } bt_bad[126];
};
```

Unused slots in the *bt_bad* array are filled with all bits set, an accepted illegal value.

The `bad144` command is invoked by giving a device type (for example, rk07, rm03, rm05, and so forth), and a device name (for example, hk0, hp1, and so forth). It reads the first sector of the last track of the corresponding disk and prints out the bad sector information. It may also be invoked giving a serial number for the pack and a list of bad sectors, and will then write the supplied information onto the same location. Note, however, that `bad144` does not arrange for the specified sectors to be marked bad in this case. This option should only be used to restore known bad sector information which was destroyed. It is necessary to reboot before the change will take effect.

If the disk is an RP06, Fujitsu Eagle, or Ampex Capricorn on a Massbus, the **–f** option may be used to mark the bad sectors as ''bad''. This can only be done safely when there is no other disk activity, preferably while running single-user. Otherwise, new bad sectors can be added only by running a formatter. Note that the order in which the sectors are listed determines which sectors used for replacements. If new sectors are being inserted into the list on a drive that is in use, care should be taken that replacements for existing bad sectors have the correct contents.

Restrictions

On an 11/750, the standard bootstrap drivers used to boot the system do not understand bad sectors, handle ECC errors, or the special SSE (skip sector) errors of RM80 type disks. This means that none of these errors can occur when reading the file /vmunix to boot. Sectors 0-15 of the disk drive must also not have any of these errors.

The drivers which write a system core image on disk after a crash do not handle errors. Thus the crash dump area must be free of errors and bad sectors.

See Also

badsect(8), format(8v)

Name

badsect – create files to contain bad sectors

Syntax

/etc/badsect bbdir sector ...

Description

The `badsect` command makes a file to contain a bad sector. Normally, bad sectors are made inaccessible by the standard formatter, which provides a forwarding table for bad sectors to the driver. For further information, see `bad144(8)`. If a driver supports the bad blocking standard it is much preferable to use that method to isolate bad blocks, since the bad block forwarding makes the pack appear perfect, and such packs can then be copied with `dd(1)`. The technique used by this program is also less general than bad block forwarding, as `badsect` can't make amends for bad blocks in the i-list of file systems or in swap areas.

On some disks, adding a sector which is suddenly bad to the bad sector table currently requires the running of the standard DEC formatter. Thus to deal with a newly bad block or on disks where the drivers do not support the bad-blocking standard `badsect` may be used to good effect.

The `badsect` command is used on a quiet file system in the following way: First mount the file system, and change to its root directory. Make a directory BAD there. Run `badsect` giving as argument the BAD directory followed by all the bad sectors you wish to add. (The sector numbers must be relative to the beginning of the file system, but this is not hard as the system reports relative sector numbers in its console error messages.) Then change back to the root directory, unmount the file system and run `fsck(8)` on the file system. The bad sectors should show up in two files or in the bad sector files and the free list. Have `fsck` remove files containing the offending bad sectors, but do not have it remove the BAD/*nnnnn* files. This will leave the bad sectors in only the BAD files.

The `badsect` command works by giving the specified sector numbers in a `mknod`(2) system call, creating an illegal file whose first block address is the block containing bad sector and whose name is the bad sector number. When it is discovered by `fsck` it will ask ''HOLD BAD BLOCK''? A positive response will cause `fsck` to convert the inode to a regular file containing the bad block.

Restrictions

If more than one sector that comprises a file system fragment is bad, you should specify only one to `badsect,` as the blocks in the bad sector files cover all the sectors in a file system fragment.

Diagnostics

The `badsect` command refuses to attach a block that resides in a critical area or is out of range of the file system. A warning is issued if the block is already in use.

See Also

bad144(8), format(8v), fsck(8)

Name

bindsetup – set up the Berkeley Internet Name Domain (BIND)/Hesiod service

Syntax

/usr/etc/bindsetup [**–c** [**–d** *directory*] **–b** *binddomain name1,IP1 name2,IP2 ...*]

Description

The `bindsetup` command sets up the Berkeley Internet Name Domain (BIND)/Hesiod service on your system and places `aliases`, `auth`, `group`, `hosts`, `networks`, `passwd`, `protocols`, `rpc`, and `services` resolution under BIND/Hesiod control. You can use this command to set up your system as a primary, secondary, slave, or caching server, or as a client.

In order to run BIND/Hesiod, your system's host name must include the BIND domain name. The BIND host name consists of the local host name plus the BIND domain name, separated by periods. For example, the BIND host name for a system whose local host name is `orange`, and whose BIND domain name is `col.ecd.com` is `orange.col.ecd.com`.

The `bindsetup` command edits the `/etc/hosts` and `/etc/rc.local` files and changes the local host name to the BIND host name, if it is not there already.

If the `bindsetup` command changes your system's host name, you should reboot the system to be sure that the change is propogated throughout the system.

Before you run `bindsetup` , your system must be established on a local area network. In addition, you must know the BIND domain name for your local area network, and whether your system will be a primary, secondary, slave, or caching server, or a client.

The `bindsetup` command asks if you want to run a Kerberos authentication server. You must already have set up Kerberos to do do. For more information, see the *Guide to Kerberos.*

You should run the `bindsetup` command as superuser and with the system in multiuser mode.

If you use the `-c` option with the respective arguments, the `bindsetup` command sets up your system as a BIND/Hesiod client non-interactively.

If you run the `bindsetup` command with no arguments, a menu is displayed giving you a choice of responses. You are then prompted for further information. Before `bindsetup` exits, it lists the files that have been updated.

Once BIND/Hesiod is installed on a machine, it cannot be used until the `/etc/svc.conf` file is modified to contain BIND entries on the desired database lines. The `bindsetup` command reminds a user to run `/usr/etc/svcsetup` or edit the `/etc/svc.conf` file manually.

Options

–c Sets up your system as a BIND/Hesiod client according to the following arguments you supply on the command line:

–d *directory* This option and argument are required if you are setting up a diskless client from the diskless server. The *directory* is the full path name of the root directory for your system (a diskless client) on the diskless server. The following is an example of a root directory for a diskless client named `orange`:

```
/dlclient0/orange.root
```

–b *binddomain*
This is the name of the BIND domain on which your system will be a BIND client. For example, `cities.us` is a sample BIND domain name.

name,IP This is the host name and the IP address of the BIND server on the domain, for example `foobar,128.11.22.33`. You can specify one or more BIND server by listing more *name,IP* arguments, each separated by a space.

Files

`/etc/hosts`	List of locally maintained host names and IP addresses
`/etc/rc.local`	Startup commands pertinent to a specific system
`/etc/svc.conf`	Database name with the selected naming services
`/etc/hesiod.conf`	Hesiod configuration file
`/etc/kerb.conf`	List of Kerberos servers
Default BIND Files:	
`/var/dss/namedb`	BIND server data file directory
`/var/dss/namedb/named.boot`	BIND server boot file
`/var/dss/namedb/named.ca`	BIND server cache file
`/var/dss/namedb/named.local`	BIND server local host reverse address host file
`/var/dss/namedb/hosts.db`	BIND primary server hosts file
`/var/dss/namedb/hosts.rev`	BIND primary server reverse address hosts file
`/etc/resolv.conf`	BIND data file

See Also

nslookup(1), hesiod(3), hesiod.conf(5), svc.conf(5), svcsetup(8), named(8), krb.conf(5), resolv.conf(5)
Guide to the BIND/Hesiod Service
Guide to Kerberos

Name

biod – Start NFS asynchronous block I/O daemons

Syntax

/etc/biod [*ndaemons*]

Description

The `biod` daemon starts the specified number of asynchronous block I/O daemons. The *ndaemons* argument tells `biod` how many asynchronous block I/O daemons to start. The `biod` daemon is only useful to NFS clients. This command is used by NFS clients to perform read-ahead and write-behind of remote file system blocks. Like the `nfsd`(8nfs) daemon, `biod` is normally invoked at boot time via the `/etc/rc.local` file.

Examples

```
/etc/biod 2    /* start two biod daemons */
```

See Also

exports(5nfs), mountd(8nfs), nfsd(8nfs)

Name

bootpd – Server to help boot diskless clients

Syntax

/etc/bootpd [**–d**] [**–i**]

Description

The bootpd server is for the Internet BOOTP protocol (a UDP-based protocol). This allows a diskless machine to find out its Internet address, the address of a bootserver, and the name of a file to boot.

The bootpd server is either started from /etc/rc.local, or from inetd. If bootpd is started from inetd, the **–i** flag must be supplied by /etc/inetd.conf. The bootpd server reads its configuration file, /etc/bootptab, when it starts up. When a new request arrives, bootpd checks to see if the file has been modified, and if so, reads it again.

If started by inetd, bootpd waits until no new requests arrive for one minute. This limits the overhead of restarting the daemon without tying up a process slot when nothing is happening. The following is an example of the format of the configuration file:

```
#
# /etc/bootptab:  database for bootp server (/etc/bootpd)
#
# Blank lines and lines beginning with '#' are ignored.
#
# home directory

/usr/local/bootfiles

# default bootfile

defaultboot

# end of first section

%%

#
# The remainder of this file contains one line per client
# interface with the information shown by the table headings
# below. The host name is also tried as a suffix for the
# bootfile when searching the home directory (that is,
# bootfile.host)
#
# host       htype haddr          iaddr         bootfile
#

hostx        1 02:60:8c:06:35:05 99.44.0.65    ultrix
hosty        1 02:07:01:00:30:02 99.44.0.65    vms
hostz        1 02:60:8c:00:77:78 99.44.0.03    lps40
node1        1 02:60:8c:00:99:47 99.44.0.01    tops20
```

The first two lines specify the home (default) directory and the default bootfile, respectively. A line starting with two percent signs (%%) separates these first lines from the host information table, which contains an entry for each bootable host.

You should start with a configuration file similar to this and edit the host entries to correspond to your local systems. The host field does not have to be a formal host name; it is used for identification in the log file and also as a possible extension to the bootfile name.

The `htype` is always 1 and corresponds to the hardware type assigned Ethernet by the Assigned Numbers RFC. The `haddr` field can use a period (.), a hyphen (-), or a colon (:) as separators. The `bootfile` entry is the file used if the client does not know the name of the file it wants to boot. This is frequently the case when a diskless workstation is booted.

The `bootpd` server logs interesting events using `syslog`.

Options

–d	Logs all requests and indicates what responses are made.
–i	If `bootpd` is started from `inetd`, the **–i** flag must be supplied by `/etc/inetd.conf`.

Files

`/etc/bootptab`	Configuration file

See Also

tftpd(8), inetd(8)

catman (8)

Name

catman – create the cat files for the manual

Syntax

/etc/catman [**–p**] [**–n**] [**–w**] [*sections*]

Description

The `catman` command creates the preformatted versions of the on-line manual from the `nroff` input files. Each manual page is examined and those whose preformatted versions are missing or out of date are recreated. If any changes are made, `catman` will recreate the `/usr/lib/whatis` database.

If there is one parameter not starting with a minus sign (–), it is taken to be a list of manual sections to look in. The following example causes updating to occur to manual sections 1, 2, and 3.

```
catman 123
```

Options

–n	Prevents creations of `/usr/lib/whatis`.
–p	Prints what would be done instead of doing it.
–w	Causes only the `/usr/lib/whatis` database to be created. No manual reformatting is done.

Files

`/usr/man/man?/*.*`	Raw (nroff input) manual sections
`/usr/man/cat?/*.*`	Preformatted manual pages
`/usr/lib/makewhatis`	Commands to make whatis database

See Also

man(1)

Name

ccr – remote console carrier requester

Syntax

ccr [*options*] *node*

Description

The `ccr` command establishes a logical connection between your ULTRIX system and the console carrier server on a remote system. `ccr` enables your terminal to act as the console for a remote unattended system. For example, your terminal can act as the console for a Digital Ethernet Communications Server (DECSA) and its resident software. The *node* is the name or address of the target node. A node name consists of from one to six alphanumeric characters. A node address consist of two decimal integers (n.n), where the first indicates the network number (1-63), and the second indicates the node address (1-1023).

You can use `ccr` to force a crash if a server node becomes unresponsive. To determine how to force a crash, see the documentation for the respective server product.

The requirements for using `ccr` are as follows:

- The host node (that is, your local ULTRIX node) and the remote node must be on the same Ethernet.
- If your server product is a DECSA, the console carrier server image (plutocc.sys) and its loader file (plutowl.sys) must be located in `/usr/lib/dnet` on your ULTRIX node. The pluocc.sys and plutowl.sys files are not need, nor is any loading done for other servers. For more details, see the installation guide for the particular server product.

<CTRL/D> exits from console carrier mode and terminates `ccr`.

Options

–c Uses the specified circuit to connect to the target node.

–h Uses the specified address (next argument) as the Ethernet address of the target node.

–n Uses the next argument as the target node ID.

–p Uses the specified service password (next arguments) in accessing the target node.

Examples

```
# /etc/ccr -c qna-0 -n dallas <RET>
ccr: Remote console reserved
       .
       .
       .
<CTRL/D>
ccr: Remote console released
#
```

Restrictions

You must have superuser privileges to run `ccr`.

Diagnostics

The `ccr` command can return the following diagnostic messages:

ccr: Remote console reserved
The `ccr` command has successfully connected to the remote console server and your terminal is now capable of acting as a console for the remote node.

ccr: Remote console released
Your connection with the remote console server has been terminated and you are no longer in console carrier mode.

ccr: Remote console already in use
The remote console server that you are attempting to connect to is already reserved by another user.

ccr: Permission denied
You do not have the necessary privileges to run `ccr`. (You must be a superuser.)

ccr: Hardware address required
The `ccr` command is unable to locate the hardware address of the remote node to which you are attempting to connect. A remote node's hardware address must be defined either in the `ccr` command line, or in its nodes database entry. (Nodes database entries are defined with the `addnode` command.)

ccr: No node entry in database
The `ccr` command does not recognize the remote node to which you are attempting to connect, since the *node-id* that you specified is not defined in the nodes database. (Nodes database entries are defined with the `addnode` command.)

Files

`/usr/lib/dnet/plutocc.sys`
Console carrier server image

`/usr/lib/dnet/plutowl.sys`
Console carrier server loader

See Also

addnode(8), getnode(8), load(8), remnode(8), trigger(8)
Guide to Local Transport Servers

Name

chown – change owner and, optionally, group

Syntax

/etc/chown [**–fR**] *owner*[*.group*]*file*...

Description

The `chown` command changes the owner and, optionally, group for one or more files and directories. The value for *file* can be a full or partial path. The value for *owner* can be either a decimal UID or a login name found in the password file. The value for *group* can be either a decimal GID or a group name found in the group file.

The user invoking `chown` must be either the superuser or the owner of all specified files and, if *group* is specified, must belong to the specified group.

Options

–f Inhibits display of errors that are returned when `chown` cannot change the owner or group of the specified files.

–R Causes `chown` to recursively descend any directories subordinate to *file* and to set the owner, group, or both for each file encountered. When symbolic links are encountered, `chown` changes the owner and group for the link file itself but does not traverse the path associated with the link. The `-R` option is useful only when *file* is a directory that is not empty.

Examples

Change the owner of `myfile` to ecbell:

```
/etc/chown ecbell myfile
```

Change the owner of `myfile` to craig and group of `myfile` to admin:

```
/etc/chown craig.admin myfile
```

Change the owner to richart and group to eng for the directories `projecta` and `projectb` and for all files and directories on any levels subordinate to `projecta` and `projectb`:

```
/etc/chown -R richart.eng projecta projectb
```

Files

`/etc/passwd`

`/etc/group`

`/etc/yp/src/passwd`

`/etc/yp/src/group`

See Also

chgrp(1), chown(2), group(5), group(5yp), passwd(5), passwd(5yp)

Name

chpt – change a disk partition table

Syntax

/etc/chpt [**–a**] [**–d**] [**–q**] [**–v**] [[**–p***x* *offset size*] ...] *device*

Description

The `chpt` command lets you alter the partition sizes of a disk pack. Using `chpt`, you can tailor your system disks and their partitions to suit your system's individual needs.

If you want to create a file system on a partition that has been modified, you must use `newfs`(8).

The standard procedure to change a partition table is:

1. Look at the current partition table using the **–q** option.
2. If a file system does not exist on the *a* partition, create one using the `newfs`(8) command.

   If a file system exists on the *a* partition but does not contain a partition table in its superblock, copy the partition table from the driver to the superblock using the `chpt` command with the **–a** option.
3. Change the partition offsets and sizes using the **–p***x* option. You can change all the partitions for one pack on one command line.

The device must be either the *a* or *c* partition of the raw device, depending upon where the file system resides. For example, if the file system resides in the *a* partition of an RM05 in drive 0, *device* is rhp0a.

A file system must exist on the *a* or *c* partition of the pack. If you do not have a file system there, create one using `newfs`(8).

Options

–a Copies the partition table in the device driver to the disk pack.

–d Copies the default partition table to the disk pack and to the current partition table in the driver. The default partition table is the table that was built with the disk driver.

–q Runs `chpt` without modifying the partition tables. This prints the partition table of the specified disk pack. It prints the default partition table in the driver if there is no partition table on the disk pack.

–v Prints verbose messages showing the progress of `chpt`.

–p*x* Changes the parameters of partition *x* on the disk pack to the specified *offset* and *size*. *x* is the partition you are modifying (a, b, c, d, e, f, g, or h). *Offset* is the new beginning sector, and *size* is the new total number of sectors of the partition being modified.

Examples

This example shows how to change the partition table on an RM05 disk pack in drive 1. The commands in this example change the the size of the *h* partition to include the *g* partition. Comments are in parenthesis to the right of commands.

```
% chpt -q /dev/rhp1a                  (view partition table)
/dev/rhp1a
No partition table found in superblock...
using default table from device driver.
Current partition table:
partition       bottom        top       size     overlap
    a                0      15883      15884     c
    b            16416      49855      33440     c
    c                0     500383     500384     a,b,d,e,f,g,h
    d           341696     357579      15884     c,g
    e           358112     414047      55936     c,g
    f           414048     500287      86240     c,g
    g           341696     500287     158592     c,d,e,f
    h            49856     341201     291346     c
%
```

NOTE

In all of the tables generated by `chpt,` *bottom* is the offset (starting sector), *top* is the ending sector, and *size* is the number of sectors in the partition. The *overlap* is the other sectors that are partially or entirely included in the partition.

```
% bc                          (basic calculator)
500287-49856                  (top of g minus bottom of h)
450431
450431+1                      (add 1 because it is zero-based)
450432                        (size of new h partition)
%
```

From the query, you can see that there is no partition table in the superblock of the *a* partition. If this is because there is no file system in the *a* partition, run the `newfs` command to create one.

For this example, assume that there is a file system in the *a* partition of the disk, but the file system does not contain a partition table in its superblock. Therefore, run `chpt` with the **-a** option to copy the partition table in the driver to the superblock of the *a* partition.

```
% chpt -a /dev/rhp1a                  (add table to a partition)
%
```

Now you have a partition table to change.

```
% chpt -v -ph 49856 450432 /dev/rhp1a                    (change h)
/dev/rhp1a
New partition table:
partition        bottom   top      size     overlap
    a                0    15883    15884    c
    b            16416    49855    33440    c
    c                0   500383   500384    a,b,d,e,f,g,h
    d           341696   357579    15884    c,g,h
    e           358112   414047    55936    c,g,h
    f           414048   500287    86240    c,g,h
    g           341696   500287   158592    c,d,e,f,h
    h            49856   500287   450432    c,d,e,f,g
%
```

Caution

Changing partition tables indiscriminately can result in losing large amounts of data.

Check for file systems on all the partitions of the disk before using the **–p** option. If a file system exists whose partition may be destroyed, copy it to a backup medium. After you have changed the partitions, restore the backed up file system.

Restrictions

You must have superuser privileges to use `chpt`.

You can not shrink or change the offset of a partition with a file system mounted on it or with an open file descriptor on the entire partition.

You can not change the offset of the *a* partition.

See Also

ioctl(2), disktab(5), fsck(8), mkfs(8), newfs(8)
Guide to System Disk Maintenance

clri (8)

Name

clri – clear inodes

Syntax

/etc/clri *filesystem i-number ...*

Description

The `clri` command has been superseded by `fsck`(8) for normal file system repair work.

The `clri` command writes zeros on the i-nodes with the decimal *i-numbers* on the *filesystem.* After `clri,` any blocks in the affected file will show up as 'missing' in an `icheck`(8) of the *filesystem.*

Read and write permission is required on the specified file system device. The i-node becomes allocatable.

The primary purpose of this command is to remove a file which for some reason appears in no directory. If it is used to zap an i-node which does appear in a directory, care should be taken to track down the entry and remove it. Otherwise, when the i-node is reallocated to some new file, the old entry will still point to that file. At that point removing the old entry will destroy the new file. The new entry will again point to an unallocated i-node, so the whole cycle is likely to be repeated again and again.

Restrictions

If the file is open, `clri` is likely to be ineffective.

See Also

icheck(8)

Name

cmx – generic communication exerciser

Syntax

/usr/field/cmx [**–h**] [**–o***file*] [**–t***n*] **–l** *line-1* ...

Description

The `cmx` exerciser will write, read, and validate random data and packet lengths on a given communications line. The line under test must have a loopback connector attached to the distribution panel or the cable and the line must be disabled in the `/etc/ttys` file and a non-modem line. That is, the `ty_status` flag must be set to off.

The exerciser runs until <CTRL/C> or kill -15 *pid* is sent to the process.

A logfile is made in `/usr/field` for you to examine and then remove. If there are errors listed in the logfile, make sure that you check the `/usr/adm/syserr/syserr.<hostname>` file, because that is where the driver and kernel error messages are saved.

You must specify the **–l** flag followed by the lines to test. The *line-n* arguments identify the lines to be tested. A maximum of 32 lines can be tested at any one time. The *line-n* arguments are specified as names taken from the `/dev` directory without the letters ''tty''. For example, if the `/dev` directory lists tty03, the *line* argument is 03.

The DEVICES section lists the devices that can be tested.

Options

–h	Prints help message about this command.
–o*file*	Save output diagnostics in *file.*
–t*n*	Run time in minutes (*n*). The default is to run until a <CTRL/C> or **kill -15** *pid* is sent to the process.

Restrictions

If there is a need to run a system exerciser over an NFS link or on a diskless system there are some restrictions. For exercisers that need to write into a file system, such as `fsx`(8), the target file system must be writable by root. Also the directory, in which any of the exercisers are executed, must be writable by root because temporary files are written into the current directory. These latter restrictions are sometimes difficult to overcome because often NFS file systems are mounted in a way that prevents root from writing into them. Some of the restrictions may be overcome by copying the exerciser to another directory and then executing it.

Pseudo devices (those whose first character after tty is p, q, r, s, t, u) cannot be tested. Neither can lta devices with major #39.

Devices

Use the `file` command on `/dev/tty*` to find out which tty line corresponds to a device line number.

Examples

The following example runs the `cmx` exerciser for 60 minutes on lines 00, 13, 22, and 32.

```
% /usr/field/cmx -t60 -l 00 13 22 32
```

The following example runs the `cmx` exerciser on lines 11, 42, 45, and 76 in the background until interrupted by a <CTRL/C> or **kill -15** *pid.*

```
% /usr/field/cmx -l 11 42 45 76 &
```

See Also

Guide to System Exercisers

Name

comsat – biff server

Syntax

/etc/comsat

Description

The `comsat` command is the server process which receives reports of incoming mail and notifies users if they have requested this service. The `comsat` command is invoked by `inetd`(8c), when it detects an incoming packet on the datagram port associated with the ''biff'' service specification. For further information, see `services`(5). The `comsat` command reads the packet, which is a one line message of the form:

user@mailbox-offset

If the *user* specified is logged in to the system and the associated terminal has the owner execute bit turned on (by a ''biff y''), the *offset* is used as a seek offset into the appropriate mailbox file and the first 7 lines or 560 characters of the message are printed on the user's terminal. Lines which appear to be part of the message header other than the From:, To:, Date:, or Subject: lines are not included in the displayed message.

Restrictions

The message header filtering is prone to error.

Files

`/etc/utmp`	Information about who is logged on to which terminals

See Also

biff(1), inetd(8c)

Name

config – build system configuration files

Syntax

/etc/config [**–p**] [**–s**] *config_file*

Description

The `config` command builds a set of system configuration files from a short file which describes the sort of system that is being configured. It also takes as input a file which tells `config` what files are needed to generate a system. This can be augmented by a configuration specific set of files that give alternate files for a specific machine. (See the Files section.) If the **–p** option is supplied, `config` will configure a system for profiling. You must have sources to use the **–p** option. Use the **–s** option when building a kernel from sources.

The `config` command should be run from the `conf` subdirectory of the system source (usually `/sys/conf`). The `config` command assumes that there is already a directory `../config_file` created and it places all its output files in there. The output of `config` consists of a number files: `ioconf.c` contains a description of what I/O devices are attached to the system, and `makefile` is a file used by `make`(1) in building the system; a set of header files which contain the number of various devices that will be compiled into the system; and a set of swap configuration files which contain definitions for the disk areas to be used for swapping, the root file system, argument processing, and system dumps.

After running `config,` it is necessary to run `make depend` in the directory where the new makefile was created. The `config` command reminds you of this when it completes.

If you receive other error messages from `config,` fix the errors in your configuration file and try again. If compile a system that has configuration errors, the system will fail.

Restrictions

The line numbers reported in error messages are usually off by one.

Files

`/sys/conf/mips/makefile.mips`	Generic makefile
`/sys/conf/mips`	List of common files that the system is built from
`/sys/conf/mips/files.mips`	List of machine specific files
`/sys/conf/mips/devices.mips`	Name to major device mapping file
`/sys/conf/mips/filesystems`	List of known file systems

See Also

The Syntax portion of each device in Section 4 of the *ULTRIX Reference Pages*
''Building 4.2BSD UNIX System with Config,'' *ULTRIX Supplementary Documents, Volume 3: System Manager*

config(8)

Name

config – build system configuration files

Syntax

/etc/config [–p] [–s] *config_file*

Description

The `config` command builds a set of system configuration files from a short file that describes the sort of system that is being configured. It also takes as input a file that tells `config` what files are needed to generate a system. This can be augmented by a configuration specific set of files that give alternate files for a specific machine. (See the Files section.) If the **–p** option is supplied, `config` will configure a system for profiling. You must have sources to use the **–p** option. Use the **–s** option when building a kernel from sources. For further information, see `kgmon`(8) and `gprof`(1).

The `config` command should be run from the `conf` subdirectory of the system source (usually `/sys/conf`). The `config` command assumes that there is already a directory `../config_file` created and it places all its output files in there. The output of `config` consists of a number files: `ioconf.c` contains a description of what I/O devices are attached to the system, `ubglue.s` contains a set of interrupt service routines for devices attached to the UNIBUS, and `makefile` is a file used by `make`(1) in building the system; a set of header files which contain the number of various devices that will be compiled into the system; and a set of swap configuration files which contain definitions for the disk areas to be used for swapping, the root file system, argument processing, and system dumps.

After running `config,` it is necessary to run `make`depend in the directory where the new makefile was created. The `config` command reminds you of this when it completes.

If you get any other error messages from `config,` you should fix the problems in your configuration file and try again. If you try to compile a system that had configuration errors, you will likely meet with failure.

Restrictions

The line numbers reported in error messages are usually off by one.

Files

`/sys/conf/makefile.vax`	Generic makefile for the VAX
`/sys/conf/files`	List of common files system is built from
`/sys/conf/files.vax`	List of VAX specific files
`/sys/conf/devices.vax`	Name to major device mapping file for the VAX
`/sys/conf/files.ERNIE`	List of files specific to ERNIE system

`/sys/conf/filesystems`
List of known file systems

See Also

The Syntax portion of each device in Section 4.
ULTRIX Supplementary Documents, Volume 3: System Manager

conflict(8mh)

Name

conflict – search for alias/password conflicts

Syntax

/usr/new/lib/mh/conflict.8mh [–mail *name*] **[–search** *directory*] **[aliasfiles...]**
[–help]

Description

The `conflict` program checks to see if the interface between `MH` and transport system is in good shape. It also checks for maildrops in `/usr/spool/mail` which do not belong to a valid user. It assumes that no user name will start with a dot (.) and thus ignores files in `/usr/spool/mail` which begin with a dot (`.`). It also checks for entries in the `group(5)` file which do not belong to a valid user, and for users who do not have a valid group number. In addition, duplicate users and groups are noted.

The `conflict` program should be run under `cron`, or whenever system accounting takes place. Note that `aliasfiles` defaults to `/usr/new/lib/mh/MailAliases`

Options

-mail *name*	Sends the results to the specified *name*. Otherwise, the results are sent to the standard output.
-search *directory*	Searches directories other than `/usr/spool/mail` and to report anomalies in those directories. The **–search** *directory* switch can appear more than one time in an invocation to `conflict`.

Files

```
/usr/new/lib/mh/mtstailor
/etc/passwd
/etc/group
/usr/new
/mh/mhmail
/usr/spool/mail
```

See Also

mh-alias(5mh), cron(8)

Name

crash – examine system images

Syntax

/etc/crash [*system*] [*namelist*]

Description

The `crash` utility is an interactive program that lets you examine the core image of the operating system. This utility has facilities for interpreting and formatting the various control structures in the system and certain miscellaneous functions that are useful when perusing a dump.

The arguments to the `crash` utility are the file name where the *system* image can be found and a *namelist* file to be used for symbol values.

The default values are `/dev/mem` and `/vmunix`; hence, the `crash` utility specified without arguments can be used to examine an active system. If a *system* image file is given, it is assumed to be a system core dump and the default process is set to be that of the process running at the time of the crash. This is determined by a value stored in a fixed location by the dump mechanism.

Commands

Use the following input to the `crash` utility:

command [*options*] [*structures*]

If used, *options* modify the format of the printout. If a specific structure element is not specified, all valid entries are used. For example, **proc 12 15 3** prints only process table slots 12, 15, and 3, but **proc** prints the entire process table in standard format.

In general, those commands that perform I/O with addresses assume hexadecimal on 32-bit machines and octal on 16-bit machines.

The commands include the following:

user [*process table entry*] [**proc address*] [*#pid*]
Aliases: `uarea`, `u_area`, `u`.
Prints the user structure of the named process as determined by the information contained in the process table entry. If an entry number is not given, the information from the last executing process is printed. Swapped processes produce an error message since their uareas are swapped.

ufile [*process table entry*] [**proc address*] [*#pid*]
Prints the open file table for the given process.

trace [–] [*process table entry*] [**proc address*] [*#pid*]
Aliases: `t`.
Generates a kernel stack trace of a process. The process is either a process slot number, an address of process slot, or the running process. If the process is not running, the trace begins at the `pcb`. If an entry number is not given, the information from the last executing process will be printed. It is not possible to trace the executing process on a running system.

Using the – flag allows the registers and variables for each stack frame to be dumped.

stack [*process table entry*] [**proc address*] [*#pid*]
: Aliases : `s`, `stk`.
Generates a kernel stack dump of a process. This is an unformatted display of the kernel stack.

proc [**–r**] [*process table entry*] [**proc address*] [*#pid*]
: Aliases: `p`.
Formats the process table. The **–r** option causes only runnable processes to be printed.

proclock [**–r**] [*process table entry*] [**proc address*] [*#pid*]
: Shows the SMP sleep locks held by a non-running process. Same optional arguments as **proc**.

pcb [*process table entry*] [**proc address*] [*#pid*]
: Prints the process control block of the current process. The process control block is a part of the user area (VAXen only). If no entry number is given, the information from the last executing process will be printed.

ps [*process table entry*] [**proc address*] [*#pid*]
: Prints the process slots, process id's, and process names for all processes.

pcb [*process table entry*] [**proc address*] [*#pid*]
: Prints the process control block for the given process.

ppte [*process table entry*] [**proc address*] [*#pid*]
: Prints the pte's associated with the given process.

spt
: Dumps the system page table.

cmap *<page frame number>*
: Prints the memory freelist or the cmap for the given PFN.

cmap –i *index*
: Prints the core map for the given coremap index.

cmap –h *index*
: Prints the core maps for the given hash starting at index.

cmap -b *block*
: Prints the core maps hashed on the given block.

cmap –a *at*
: Prints the core map at the given address.

gnode [**–**] [*gnode table entries*] [**gnode address*]
: Aliases: `gno`, `g`.
Formats the gnode table.

gnode –maj *<major number>*
: Formats all gnodes with the given major number.

gnode –min *<minor number>*
: Formats all gnodes with the given minor number.

gnode –fs *<mount slot number>*
: Formats all gnodes for a given file system slot.

gnode –gno <*gnode number*>
Formats all gnodes with the given gnode number.

gnode –uid <*user id*>
Formats all gnodes owned by the given uid.

gnode –gid <*group id*>
Formats all gnodes owned by the given gid.

gnode –lmod <*file permissions*>
Formats all gnodes with the given protection. The command **gnode -lmod 777** finds all gnodes that allow read/write/execute permission to every one.

gnode –hmod <*file type*>
Formats all gnodes with the given file type. **gnode -hmod 2** find all gnodes that are character special files.

gnode –amod <*file modesfR*>
Formats all gnodes that match the given modes exactly. The command **gnode -amod 20777** finds all gnodes for character devices that allow read/write/execute permission to every one.

gnode –all
Displays a more extensive list of the gnodes contents.

gnode –lock
Shows the SMP lock contained in the gnode.

gfree Prints the list of all inactive gnodes.

block [**–**] [*gnode table entries*] [**gnode address*]
Prints the gnode data block addresses.

rnode [*gnode table entries*] [**gnode address*]
Alias: `v`.
Prints the associated rnode values for gnodes that are remote (via NFS).

file [*file table entries*] [**file address*]
Alias: `f`.
Formats the file table.

cred *address*
Prints the credentials at the given address.

crred *address*
Verifies the references of a credential at the given address.

crcheck Verifies all references to all credentials.

mount [–s] [*mount table entries*] [**mount address*]
Aliases: `mnt`, `m`.
Formats the mount table. The `-s` option gives a abbreviated format.

fsdata [*mount table entries*] [**mount address*]
Alias: `df`.
Prints the fsdata structure associated with a mount entry.

mntinfo [*mount table entries*] [**mount address*]
Alias: `mi`.
Prints the mntinfo data structure associate with an NFS file system.

buf [–] [*buffer headers*] [**buffer header address*]
Aliases: `hdr`, `bufhdr`.
Formats the system buffer headers. By using the – option, all buffer headers (including ones marked invalid) are printed.

bufgp [*gnode slots*] [**gnode address*]
Alias: `cache`.
Prints the buffer headers associated with the given gnode.

buflock Prints the buffer headers on the locked list.

buflru Prints the buffer headers on the lru list.

bufage Prints the buffer headers on the aged list.

bufempty
Prints the buffer headers on the empty list.

buffer [*format*] [*list of buffers*]
Alias: `b`.
Prints the data in a system buffer according to *format*. If *format* is omitted, the previous *format* is used. Valid formats include `decimal`, `octal`, `hex`, `character`, `byte`, `directory`, `gnode`, and `write`. The last creates a file in the current directory (see the Files section) containing the buffer data.

text [*text table entries*] [**text address*]
Aliases: `txt`, `x`.
Formats the text table.

ftext Alias: `freet`.
Prints the list of free texts.

callout Aliases: `calls`, `call`, `c`, `timeout`, `time`, `tout`.
Prints all entries in the callout table.

arp [–] [*address*]
Prints the arp table. The – option prints the entire table.

socket [*file slot*] [**file address*]
Prints the socket structure associated with the given file slot.

tty [–] [*process slot*] [**proc address*] [*#pid*]
Alias: `term`.
Prints the terminal structure attached to a process. The – option allows for the raw, cannonical, and output clists.

tty –clist Includes clists in display of the tty struct.

tty -addr *address*
Prints the contents of a tty structure at the specified address.

map [*map names*]
Formats the named system map structures.

nm [*symbols*]
Prints the symbol value and type as found in the *namelist* file.

ts [*text addresses*]
Finds the closest text symbols to the given addresses.

ds [*data addresses*]
: Finds the closest data symbols to the given addresses.

od [*symbol name or address*] [*count*] [*format*]
: Aliases: `dump`, `rd`.
Dumps *count* data values starting at the symbol value or address given according to *format*. Allowable formats are `octal`, `longoct`, `decimal`, `longdec`, `character`, `hex`, or `byte`.

dis *address* [*address*]
: Disassembles starting at the first address and continuing until the second address. These addresses may be symbolic (that is, syscall+33).

stat Prints useful statistics pertaining to the buffer cache, dnlc, namei translation cache, and others.

dupreq Displays the contents of the duplicate request cache. This is useful in finding out the recent history of NFS requests made to the server.

mbuf *address*
: Displays the `mbuf` chain starting at the given address.

inpcb -udp -tcp
: Displays the `inpcb` chain of the corresponding protocol, or both TCP and UDP if no protocol is specified.

client Displays the client table, which contains client handles used to initiate rpc requests.

sync Resynchronizes the proc, mount, gnode, buffer, file and other internal tables up to the current state of `/dev/kmem`. This command is useful for looking at changing values in runing kernels. However, you chould not use it when looking at `vmcore` files.

svcxprt *address*
: Prints the svcxprt structure located at address.

scs Traverses data structures in the System Communications Services (SCS) tree and displays the contents of the data structures.

scs -cb *address*
: Displays the contents of an SCS connection block data structure at the specified address.

scs -cib *address*
: Displays the contents of an SCS connection information block data structure at the specified address.

scs -pb *address*
: Displays the contents of an SCS path block data structure at the specified address.

scs -pib *address*
: Displays the contents of an SCS path information block data structure at the specified address.

scs -sb *address*
: Displays the contents of an SCS system block data structure at the specified address.

scs -sib *address*
: Displays the contents of an SCS system information block data structure at the specified address.

scsi Prints SCSI controller information.

scsi -target
: Prints SCSI target information.

scsi -devtab
: Prints SCSI devtab information.

scsi -trans Prints SCSI transfer information.

scsi -cmd Prints SCSI message or command data.

scsi -bbr Prints SCSI Bad Block Replacement data.

scsi -error Prints SCSI error information.

scsi -sii Prints SCSI SII information.

scsi -dct Prints SCSI DCT statistics.

scsi -spin Prints SCSI SPIN statistics.

scsi -all Prints all SCSIBUS information.

lock [-all] *address*
: Prints the SMP lock structure located at address. The `all` flag displays all global SMP locks.

mscp Traverses both the mscp disk and tape subsystems, and prints the data structures for class blocks, connection blocks, unit blocks, and active request blocks.

mscp -disk
: Traverses the mscp disk subsystem and prints the data structures for the mscp class block, mscp connection blocks, mscp unit blocks, and active request blocks.

mscp -tape
: Traverses the tmscp tape subsystem and prints the data structures for the tmscp class block, tmscp connection blocks, tmscp unit blocks, and active request blocks.

mscp -config
: Traverses both the mscp disk and tape subsystems, and display the system configuration. This is done by printing summary information from the class, connection, and unit blocks.

mscp -connb *address*
: Prints the contents of an mscp connection block at the specified address.

mscp -classb *address*
: Prints the contents of an mscp class block at the specified address.

mscp -unitb *address*
: Prints the contents of an mscp unitb block at the specified address.

mscp -reqb *address*
: Prints the contents of an mscp request block at the specified address.

mscp -dtable
Displays all the elements of the mscp disk unit table. Unused elements of the array will be specified as NULL.

mscp -ttable
Displays all the elements of the tmscp tape unit table. Unused elements of the array will be specified as NULL.

! Escapes to the shell.

#[*history*]
Repeats the last command. If a number is given (that is, #5), that command number is re-executed.

#h Alias: `history`,
Shows the history list.

q Exits from `crash`.

? Prints a synopsis of commands.

Aliases

There are built-in aliases for many of the *formats* as well as those listed for the commands. Some of them are:

byte	b.
character	char, c.
decimal	dec, e.
directory	direct, dir, d.
hexadecimal	hexadec, hex, h, x.
gnode	gno , g.
longdec	ld, D.
longoct	lo, O.
octal	oct, o.
write	w.

Restrictions

Many of the flags are abbreviated making them difficult to interpret. A source listing of the system header files would be helpful when using the `crash` utility.

Examing the stack of the current process on a running system and procs running at the time of a crash does not work.

Files

`/usr/include/sys/*.h`	Header files for table and structure info
`/dev/mem`	Default system image file
`/vmunix`	Default namelist file
`buf.#`	Files created containing buffer data

See Also

mount(8), nm(1), ps(1), sh(1), stty(1), pstat(8)

Name

crash – what happens when the system crashes

Description

This section explains what happens when the system crashes and shows how to analyze crash dumps.

When the system crashes voluntarily it prints a message on the console in the form:

panic: explanation

The system takes a dump on a mass storage peripheral device or the network, and then invokes an automatic reboot procedure as described in `reboot(8)`. Unless there is some unexpected inconsistency in the state of the file systems due to hardware or software failure, the system then resumes multi-user operations. If auto-reboot is disabled, the system halts at this point.

The system has a large number of internal consistency checks; if one of these fails, it prints a short message indicating which one failed.

The most common cause of system failures is hardware failure. In all cases there is the possibility that hardware or software error produced the message in some unexpected way. These messages are the ones you are likely to encounter:

IO err in push
hard IO err in swap
The system encountered an error when trying to write to the paging device or an error in reading critical information from a disk drive. Fix your disk if it is broken or unreliable.

timeout table overflow
Due to the current data structure, running out of entries causes a crash. If this happens, make the timeout table bigger.

Exception Condition
An unexpected system error has occurred. The exception types are as follows:

Mnemonic	Description
INT	External interrupt
MOD	TLB modification exception
TLBL	TLB miss exception (load or instruction fetch)
TLBS	TLB miss exception (store)
AdEL	Address error exception (load or instruction fetch)
AdES	Address error exception (store)
IBE	Bus error exception (for an instruction fetch)
DBE	Bus error exception (for a data load or store)
Sys	Sys call exception
Bp	Breakpoint exception
CpU	Coprocessor unusable exception
Ovf	Arithmeticc overflow exception

KSP not valid
This indicates either a problem in the system or failing hardware.

init died The system initialization process has exited. The only solution is the automatic reboot procedure described in `reboot`(8). Until this is done, new users cannot log in.

When the system crashes, it attempts to write an image of memory into the back end of the primary swap area. After the system is rebooted, the program `savecore`(8) runs and preserves a copy of this core image and the current system in a specified directory for later access. See `savecore`(8) for details.

To analyze a dump, you should begin by running `dbx`(1) with the `-k` flag on the core dump.

See Also

dbx(1), reboot(8), savecore(8)

Name

crash – what happens when the system crashes

Description

This section explains what happens when the system crashes and shows how to analyze crash dumps.

When the system crashes voluntarily it prints a message on the console in the form:

panic: explanation

The system takes a dump on a mass storage peripheral device, and then invokes an automatic reboot procedure as described in `reboot`(8). Unless there is some unexpected inconsistency in the state of the file systems due to hardware or software failure, the system then resumes multi-user operations. If auto-reboot is disabled on the front panel of the machine, the system halts at this point.

The system has a large number of internal consistency checks; if one of these fails, it prints a short message indicating which one failed.

The most common cause of system failures is hardware failure. In all cases there is the possibility that hardware or software error produced the message in some unexpected way. These messages are the ones you are likely to encounter:

IO err in push
hard IO err in swap
The system encountered an error when trying to write to the paging device or an error in reading critical information from a disk drive. Fix your disk if it is broken or unreliable.

timeout table overflow
Due to the current data structure, running out of entries causes a crash. If this happens, make the timeout table bigger.

KSP not valid
SBI fault
CHM? in kernel
These indicate either a problem in the system or failing hardware. If SBI faults recur, check out the hardware or call field service. Run the processor microdiagnostics to determine if the problem is caused by an unreliable processor.

machine check %x:
description

machine dependent machine-check information
Call field service.

trap type %d, code=%d, pc=%x
An unexpected trap has occurred within the system; the trap types are:

0	reserved addressing fault
1	privileged instruction fault
2	reserved operand fault
3	bpt instruction fault

4	xfc instruction fault
5	system call trap
6	arithmetic trap
7	ast delivery trap
8	segmentation fault
9	protection fault
10	trace trap
11	compatibility mode fault
12	page fault
13	page table fault

The most common traps in system crashes are trap types 8 and 9, indicating a wild reference. The code is the referenced address, and the pc at the time of the fault is printed. These problems tend to be easy to track down if they are kernel problems because the processor stops, but there are random occurrences with unpredictable causes.

init died The system initialization process has exited. The only solution is the automatic reboot procedure described in `reboot`(8). Until this is done, no new users can log in.

When the system crashes, it attempts to write an image of memory into the back end of the primary swap area. After the system is rebooted, the program `savecore`(8) runs and preserves a copy of this core image and the current system in a specified directory for later access. See `savecore`(8) for details.

To analyze a dump, you should begin by running `adb`(1) with the `-k` flag on the core dump. Normally, the command `*(intstack-4)$c` provides a stack trace from the point of the crash and this should provide a clue as to what went wrong.

See Also

adb(1), reboot(8), savecore(8)

Name

cron – clock daemon

Syntax

/etc/cron

Description

The `cron` command executes commands at specified dates and times according to the instructions in the file `/usr/lib/crontab.` Since `cron` never exits, it should only be executed once. This is best done by running `cron` from the initialization process through the file `/etc/rc.`

The `crontab` file is examined by `cron` every minute.

Files

`/usr/lib/crontab`

See Also

crontab(5), init(8)
Guide to System Environment Setup

dcheck(8)

Name

dcheck – check directory consistency

Syntax

/etc/**dcheck** [**–i** *numbers*] [*filesystem*]

Description

The `dcheck` command is obsoleted for normal consistency checking by `fsck`(8).

The `dcheck` command reads the directories in a file system and compares the link-count in each i-node with the number of directory entries by which it is referenced. If the file system is not specified, a set of default file systems is checked.

The **–i** flag is followed by a list of i-numbers; when one of those i-numbers turns up in a directory, the number, the i-number of the directory, and the name of the entry are reported.

The program is fastest if the raw version of the special file is used, since the i-list is read in large chunks.

Diagnostics

When a file turns up for which the link-count and the number of directory entries disagree, the relevant facts are reported. Allocated files which have 0 link-count and no entries are also listed. The only dangerous situation occurs when there are more entries than links. If entries are removed, so the link-count drops to 0, the remaining entries point to nothing. They should be removed. When there are more links than entries, or there is an allocated file with neither links nor entries, some disk space may be lost but the situation will not degenerate.

Restrictions

Since `dcheck` is inherently two-pass in nature, extraneous diagnostics may be produced if applied to active file systems.

Files

Default file systems vary with installation.

See Also

fs(5), clri(8), fsck(8), icheck(8), ncheck(8)

Name

dgated – Provide daemon login service via Decnet

Syntax

/etc/dgated *host*

Description

The `/etc/dgated` daemon is spawned by `dgate`(1c) on the gateway system (the intermediate ULTRIX DECnet host).

The program maintains the ownership of the master end of a pseudo terminal, operating as an intermediary between the DECnet login process and the client instantiation of the `dgate` process.

The `dgated` daemon also writes a `utmp` entry to record activity.

Diagnostics

All diagnostic messages are written through `syslog`(3).

All network ports in use.
A free pseudo terminal could not be found.

Cannot fork
A `fork` call failed.

Cannot exec /usr/bin/dlogin
The `dlogin` routine could not be found.

Files

```
/etc/utmp
/usr/adm/wtmp
```

See Also

dgate(1c), dlogin(1dn), fork(2), syslog(3)
DECnet-ULTRIX documentation

Name

dms – diskless management services utility

Syntax

/etc/dms

Description

The `dms` utility performs diskless management services. You can use `dms` to install products into diskless management services areas on a server machine and register diskless clients so that they can access those products on the server machine, rather than having each client install the products to a disk on its local machine. The server machine can be either a VAX or a RISC machine and it can serve both VAX and RISC clients.

Before you set up a diskless management services area on a server, the following software should be installed: Local Area Network (LAN); Network File System (NFS); and Maintenance Operations Protocol (MOP).

The `dms` utility performs the following functions:

a - Add Client Processor

m - Modify Client Parameters

r - Remove Client Processor

l - List Registered Clients

s - Show Products in Diskless Environments

i - Install Software

c - Create Diskless Area on Disk

k - Kernel Rebuild or Copy

To set up a diskless management services area on a server, select `dms` functions in the following order:

1. Create Diskless Area on Disk (c)

    This sets up the disk partitions needed for the two diskless management services file systems, `/dlenv?` and `/dlclient?`. (The question mark (?) represents a unique number assigned by `dms`.) The `/dlenv?` file system contains a common root area that is copied to each client when it is registered, and a `/usr` area that is shared by all the clients. The `/dlclient?` file system contains a copy of the common root area for each registered diskless client.

2. Install Software (i)

   This installs the products from the distribution media to the `/dlenv?` file system. If the product will be accessed by VAX clients, the images from the media are put under a `root?.vax` directory under the `/dlenv` file system. If the product will be accessed by RISC clients, the images from the media will be put under a `root?.mips` directory under the `/dlenv` file system. The question mark (?) represents a unique number assigned by `dms`.

To register a diskless client, select the Add Client Processor option (a). This copies the common root area in the `/dlenv?` file system, for example, `/dlenv0/root0.vax` to the diskless client's directory in the `/dlclient?` file system, for example, `/dlclient0/hostname.root`. The server's `/etc/exports` file is updated to allow the client to access its individual root and the shared `/usr` area.

Each diskless client's area also contains a client parameter file, `/etc/dlparam`.

You must use `dms` interactively to set up a diskless environment or to modify client parameters.

There are several functions you can use either interactively or from the command line:

- Add Client Processor
- Show Products in Diskless Environments
- Remove Client Processor
- Kernel Rebuild

When the first diskless client is registered for a diskless management services area, a database file, `/usr/diskless/dmsdb`, is created on the server machine. You can use this file and the command line to manage groups of diskless clients.

Files

```
/usr/diskless/dmsdb
/etc/dlparam
```

See Also

exports(5nfs), addnode(8), setld(8)
Guide to Diskless Management Services
Guide to Server Setup

doconfig (8)

Name

doconfig – a program to aid system configuration

Syntax

/etc/doconfig [**–c** *config_file*] [**–e** *ed_script*]

Description

The `doconfig` shell script modifies a copy of the GENERIC configuration file for a new ULTRIX system kernel. The script prompts you for the system name. If the system name does not currently exist, a system configuration file with that system name is built.

Whether the configuration file is built or not, the script then allows you to edit, configure, and build until both the configuration and the kernel build run without errors. Additional runs of `doconfig` can be made to tune the configuration further.

When new hardware is added to a system, the configuration file should be updated to reflect the new system configuration. When updating an existing configuration file or creating a new configuration file with `doconfig`, the system must be operating the generic kernel or new hardware may not be found. To successfully complete the `doconfig` process, follow these steps:

1. Save the running `vmunix` as `vmunix.old`.
2. Move `/genvmunix` to `/vmunix`.
3. Reboot the system to single user mode.
4. Check file systems.
5. Mount the `/usr` file system.
6. Run the `doconfig` program. (When execution is complete, make a note of the message `doconfig` prints showing the path and location of the new `vmunix`.)
7. Move `/vmunix` to `/genvmunix`.
8. Copy the new `vmunix` (from the message noted above) to `/vmunix`.
9. Reboot the system.

Options

–c The name of the existing configuration file should be supplied without specifying the pathname. This file should exist in the `/sys/conf/mips` directory. A new kernel is built using the specified configuration file.

–e When specified, the configuration file is edited by the ed script before building the new kernel.

Files

`/tmp/`*SYSTEMNAME*
`/sys/MIPS/`*SYSTEMNAME*
`/sys/conf/mips/`*SYSTEMNAME*

See Also

ed(1), config(8)
Installation Guide
Guide to System Configuration File Maintenance

dp(8mh)

Name

dp – parse dates RFC 822-style

Syntax

/usr/new/lib/mh/dp [–form *formatfile*] **[–format** *string*] **[–width** *columns*] **dates [–help]**

Description

The `dp` command parses dates according to the ARPA Internet standard. It also understands many non-standard formats, such as those produced by TOPS–20 sites and some UNIX sites using `ctime`(3). It is useful for seeing how `MH` will interpret a date.

The `dp` program treats each argument as a single date, and prints the date out in the official RFC 822–format. Hence, it is usually best to enclose each argument in double-quotes for the shell.

To override the output format used by `dp`, the `-format string` or `-format file` switches are used. This permits individual fields of the address to be extracted with ease. The string is simply a format string and the file is simply a format file. See `mh-format`(5mh) for the details.

The default format string used by `dp` is:

```
%<(nodate{text})error: %{text}%|%(pretty{text})%>
```

When an error is detected this prints `error:` and the date in error. Otherwise, output the RFC 822–proper format of the date. The width value `-width` defaults to the width of the terminal.

The argument to the `-format` switch must be interpreted as a single token by the shell that invokes `dp`. Place the argument to this switch inside double quotes.

Files

`$HOME/.mh_profile`
User profile

See Also

ap(8mh)
Standard for the Format of ARPA Internet Text Messages (RFC 822)

Name

drtest – standalone disk test program

Description

The `drtest` program is a standalone program used to read a disk track by track. It was primarily intended as a test program for new standalone drivers, but has shown useful in other contexts as well, such as verifying disks and running speed tests. For example, when a disk has been formatted (by `format`(8v)), you can check that hard errors have been taken care of by running `drtest`. No hard errors should be found, but in many cases quite a few soft ECC errors will be reported.

While `drtest` is running, the cylinder number is printed on the console for every 10th cylinder read.

Examples

A sample run of `drtest` is shown below. In this example (using a 750), `drtest` is loaded from the root file system; usually it will be loaded from the machine's console storage device. Boldface means user input. As usual, the number sign (#) and the at sign (@) can be used to edit input.

```
>>>B/3
%%
loading hk(0,0)boot
Boot
: hk(0,0)drtest
Test program for stand-alone up and hp driver

Debugging level (1=bse, 2=ecc, 3=bse+ecc)?
Enter disk name [type(adapter,unit), e.g. hp(1,3)]? hp(0,0)
Device data: #cylinders=1024, #tracks=16, #sectors=32
Testing hp(0,0), chunk size is 16384 bytes.
(chunk size is the number of bytes read per disk access)
Start ...Make sure hp(0,0) is online
 ...
(errors are reported as they occur)
 ...
(...program restarts to allow checking other disks)
(...to abort halt machine with ^P)
```

Diagnostics

The diagnostics are intended to be self explanatory. Note, however, that the device number in the diagnostic messages is identified as *typeX* instead of *type(a,u)* where *X* = a*8+u, for example, hp(1,3) becomes hp11.

See Also

bad144(8), format(8v)

Name

dskx – generic disk exerciser

Syntax

/usr/field/dskx [*options*] **–r***dev*
/usr/field/dskx [*options*] **–p***devpart*
/usr/field/dskx [*options*] **–c***dev*

Description

The `dskx` exerciser tests the disk drives on your system. The exerciser has three main options which include read only, write/read/validate data on a partition, and write/read/validate data of a disk.

The exerciser does random seeks and reads of random block sizes and random seeks, writes, reads, and validations of random data patterns of random block sizes. The exerciser will run until <CTRL/C> or kill –15 *pid* is sent to the process.

A logfile is made in `/usr/field` for you to examine and then remove. If there are errors in the logfile, make sure you check the `/usr/adm/syserr/syserr.<hostname>` file, because that is where the driver and kernel error messages are saved.

CAUTION: Both the **–c** and **–p** options of the `dskx` exerciser destroy data on the disk. Use extreme caution before using them on any non-scratch media. If you are unsure of what data is on the disk, contact your system manager before running `dskx` with either of these options.

Arguments

One of the following function flags and arguments must be specified.

–r*dev* Performs a random read-only test on all partitions except the **c** partition. The *dev* argument can be a raw or buffered device name and number. For example, rhp0, ra3, hk1.

–p*devpart* Writes, reads, and validates data from the device name and number specified by *dev* and the partition specified by *part*. The *dev* argument can be a raw or buffered device name and number. For example, rhp0, ra3, hk1. The *part* argument can be any valid partition from **a** – **h**.

CAUTION: Be careful when exercising partitions that overlap other partitions, as you may inadvertently destroy data on a partition that you do not want to test. You can use the **–q** option of the `chpt(8)` command to see what partitions overlap on the device on your system.

–c*dev* Writes, reads, and validates data from all partitions except the **c** partition on the device specified by *dev*. The *dev* argument can be a raw or buffered device name and number. For example, rhp0, ra3, hk1.

Options

The dskx options are:

–h	Prints help message for the dskx command.
–o*file*	Saves output diagnostics in *file*.
–t*n*	Specifies the run time in minutes (*n*). The default is to run until the process receives a <CTRL/C> or `kill -15` *pid*.
–d*m*	Prints statistics every *m* minutes.

Restrictions

If there is a need to run a system exerciser over an NFS link or on a diskless system there are some restrictions. For exercisers that need to write into a file system, such as fsx(8), the target file system must be writable by root. Also the directory, in which any of the exercisers are executed, must be writable by root because temporary files are written into the current directory. These latter restrictions are sometimes difficult to overcome because often NFS file systems are mounted in a way that prevents root from writing into them. Some of the restrictions may be overcome by copying the exerciser to another directory and then executing it.

Examples

The following example exercises RA disk unit 1, for 60 minutes in the background:

```
% /usr/field/dskx -t60 -cra1 &
```

The following example exercises raw HP disk unit 0, partition d, until <CTRL/C> or **kill –15** *pid*:

```
% /usr/field/dskx -prhp0d
```

See Also

Guide to System Exercisers

Name

dump – create file system dump

Syntax

/etc/dump [*key* [*argument...*] *filesystem*]

Description

The dump command copies all files changed after a certain date from a specified *filesystem* to a file, a pipe, magnetic tapes, or disks. The *key* specifies the date and other options to be used by dump.

Dumping a filesystem requires operator attention. An operator must intervene when the end of a tape or disk is reached, when the end of the dump occurs, or when an unrecoverable input disk read error occurs (if more than 32 read errors occur). In addition to alerting all operators in the operator group, dump interacts with the operator at the control terminal when dump can no longer proceed, or if something is grossly wrong. All questions that dump poses must be answered by typing yes or no.

Because a full dump uses considerable system time, dump checkpoints itself at the start of each tape or disk volume. If writing that volume fails, dump asks the operator to restart from the checkpoint after the present tape or disk has been replaced.

The dump utility reports to the operator periodically, giving usually low estimates of the number of blocks to write, the number of tapes or disks the dump will take, the time to completion, and the time until the tape or disk must be changed. The output is verbose, so that others know that the terminal controlling dump is busy, and will be for some time.

This utility supports EOT handling which allows the use of multiple media. The utility prompts for the next volume when it encounters the end of the current volume.

Options

With the dump command, you specify a string of one or more of the options described below. If no options are specified, the *key* **9u** is assumed.

0–9 Specifies the dump level. All files that were modified since the last date stored in the file /etc/dumpdates for the same filesystem at lesser levels will be dumped. If no date is determined by the level, the beginning of time is assumed. Thus, the level **0** causes the entire filesystem to be dumped.

B Indicates that the next *argument* is a number that specifies the size, in 1024-byte blocks, of a storage medium, such as a diskette or removable disk cartridge. See the first example.

d Indicates that the density of the tape, expressed in bits per inch, is taken from the next *argument*. This density is used in calculating the amount of tape used per reel. The default density is 1600 bpi.

f Places the dump on the file or device specified by the next *argument*. If the name of the file is –, dump writes to standard output. The default dump device is `/dev/rmt0h`.

n Notifies, by means similar to a `wall`(1) command, all users in the group operator when `dump` needs operator attention.

S Prints output file size in bytes, or number of volumes for devices. See the third example.

s Indicates that the next *argument* specifies the size of the dump tape, in feet. When the specified size is reached, `dump` waits for the reel to be changed. The default tape size is 2300 feet.

u Writes the date of the beginning of the dump on the file `/etc/dumpdates` if the `dump` completes successfully. This file records a separate date for each filesystem and each dump level. The format of `/etc/dumpdates` consists of one free format record per line: filesystem name, increment level and `ctime`(3) format dump date. The superuser can carefully edit `/etc/dumpdates` to change any of the fields.

W Tells the operator which file systems need to be dumped. This information is taken from the files `/etc/dumpdates` and `/etc/fstab`. The **W** option causes `dump` to print out, for each file system in `/etc/dumpdates`, the most recent dump date and level, and highlights those file systems that should be dumped. If the **W** option is used, all other options are ignored, and `dump` exits immediately.

w Lists only those filesystems that need to be dumped.

Examples

This example dumps the filesystem `/dev/ra0a` to RX50 diskettes. The `B` option is needed when running `restore`(8) to read this dump.

```
dump 9Bf 400 /dev/rra2a /dev/ra0a
```

This example dumps the filesystem `/usr/users` to a 6250 bpi tape on a TU78 tape drive:

```
dump 0undf 6250 /dev/rmt?h /usr/users
```

This example reports number of bytes to be output for a level 0 dump of the root file system. Please note: the file test is not created.

```
dump 0Sf test /
```

Restrictions

The `dump` programs returns a 1 on successful completion.

Sizes are based on 1600-bpi blocked tape.

Anything fewer than 32 read errors on the filesystem are ignored.

Each reel requires a new process, so parent processes for reels already written remain until the entire tape is written.

Files

`/dev/rrp1g`	Default filesystem to dump from
`/dev/rmt0h`	Default tape unit to dump to
`/etc/dumpdates`	Dump date record
`/etc/fstab`	Dump table: file systems and frequency
`/etc/group`	Operator group definition
`/dev/tty`	Required for user interface

See Also

dump(5), fstab(5), opser(8), restore(8), rrestore(8c)
Guide to Backup and Restore

Name

dumpfs – dump file system information

Syntax

dumpfs *filesys*|*device*

Description

The `dumpfs` command prints out the super block and cylinder group information for the file system or special device specified. The listing is very long and detailed. This command is useful mostly for finding out certain file system information such as the file system block size and minimum free space percentage.

See Also

disktab(5), fs(5), fsck(8), newfs(8), tunefs(8)

Name

edauth – edit user auth entry

Syntax

edauth *username*

Description

The edauth command is an authorization editor. edauth creates a temporary file with an ASCII representation of the current auth database entry for the user specified by *username* and then invokes an editor on the file. You can then modify the user's auth fields. Upon leaving the editor, edauth reads the temporary file and modifies the binary database to reflect the changes made. If there are errors in the temporary file edauth will allow the user to resume editing the file to fix them.

The editor invoked is ed(1), unless the environment variable EDITOR specifies otherwise. Here is an example of the temporary file produced by edauth:

```
uid = 268
password = MXP3BnKLEWW960BEJc9DbHb6
passlifemin = 1 hour
passlifemax = 60 days
passmod = 12/20/89 - 10:24:38
authmask = login,change_password,enter_password
fail_count = 0
audit_id = 268
audit_control = or
audit_syscalls = creat,unlink
audit_tevents = login:0:1
```

Each field of the auth entry is represented as a keyword followed by an equals sign. The value part of the field may be an integer, a string, a time specification, a date, or a comma-separated list of value keywords. The effect of the field is described in auth(5).

The uid, fail_count, and audit_id fields expect integer values.

The password field is a string containing the encrypted password. One way of disabling an account is to set this to a non-empty string less than 24 characters in length such as 'nologin'.

The passlifemin and passlifemax fields specify the password expiration information. They may contain an integer specifying seconds, or a combination of scaled values. The units recognized for scaling are seconds, minutes, hours, and days. Only the first letter of the unit need be supplied. A passlifemax of one day, one hour and five minutes could be specified as any of:

```
passlifemax = 1 day 1 hour 5 minutes
passlifemax = 25 h 5 m
passlifemax = 90300 seconds
passlifemax = 90300
```

in addition to other combinations.

The passmod field is a date. It is specified in the same format as the default output of the ULTRIX date(1) command. The time portion is optional and defaults to the beginning of the day.

The `authmask`, `audit_syscalls`, and `audit_tevents` fields expect a comma-separated list of value tokens. For `authmask` this is zero or more of `login`, `change_password`, and `enter_password`. For the audit information this corresponds to the name of the audit event. See the `auditmask(8)` manpage for more information on audit events.

The `audit_control` field may be one of `or`, `and`, or `off`. See the `audcntl(2)` manpage for more information on the affect of these values.

Restrictions

Only the superuser can edit `auth` entries.

Changing the `auth` entry will not affect the uid and audit information of exisiting login sessions.

If the uid field of the entry is changed the mapping to the `/etc/passwd` file will be affected. Changes to the passwd file will probably be necessary.

Diagnostics

Various messages about incorrect input. All are self-explanatory.

Files

`/etc/auth.[dir,pag]`	Contains all authorization information
`/etc/passwd`	Maps usernames to UIDs

See Also

audcntl(2), auth(5), auditmask(8), getauth(8), vipw(8)
Security Guide for Administrators

edquota(8)

Name

edquota – edit user quotas

Syntax

edquota [**–p** *proto-user*] *users...*

Description

The `edquota` command is a quota editor. You can specify one or more users on the command line. For each user, `edquota` creates a temporary file with an ASCII representation of the current disc quotas for that user and an editor is then invoked on the file. You can then modify the quotas, add new quotas, and so forth. Upon leaving the editor, `edquota` reads the temporary file and modifies the binary quota files to reflect the changes made.

If a valid quota file does not exist, `edquota` refers the user to `quotacheck`(8). The user then has the option of either supplying the command `quotacheck -f filesystem`, which automatically creates the necessary quota file, or creating an empty quota file by various other means.

If the `-p` option is specified, `edquota` duplicates for each user, the quotas of the prototypical user specified. This is the usual mechanism used to initialize quotas for groups of users.

The editor invoked is `vi`(1), unless the environment variable EDITOR specifies otherwise.

Only the superuser can edit quotas.

Diagnostics

Various messages about inaccessible files. Other messages are self-explanatory.

Files

quotas	At the root of each file system with quotas
`/etc/fstab`	Used to find file system names and locations

See Also

quota(1), quota(2), quotacheck(8), quotaon(8), repquota(8)
''Disk Quotas in a UNIX Environment,'' *ULTRIX Supplementary Documents, Volume 3: System Manager*

Name

elcsd – error logging daemon

Syntax

/etc/elcsd

Description

The `elcsd` daemon logs hardware and system-related error packets, or error messages, from the kernel errorlog buffer to the file `syserr.`<*hostname*>. Hostname is the name of the local system. At system startup, the `elcsd` daemon is invoked out of `/etc/rc`. For further information, see `rc`(8). The `elcsd` daemon must be running whenever the system is in multiuser mode.

To invoke the `elcsd` daemon in single-user mode, use the command with the **–s** option.

You can configure the `elcsd` daemon to log error packets locally from a remote ULTRIX host, or you can configure the `elcsd` daemon to send local error packets to a remote system. Do this by changing the entries in the `elcsd.conf` file. For further information, see `elcsd.conf`(5). The daemon uses an internet datagram socket to log information across systems.

Parameters related to `elcsd,` such as the desired location for the errorlog file or files, as well as instructions for local and remote logging, can be specified in the error logging configuration file, `/etc/elcsd.conf`. For further information, see `elcsd.conf`(5).

Restrictions

You must have superuser privileges to invoke the `elcsd` daemon.

Diagnostics

The `elcsd` daemon logs general status and error messages to `/usr/adm/elcsdlog` in multiuser mode. This file is purged whenever the daemon is restarted.

Files

```
/etc/elcsd.conf
/usr/adm/elcsdlog
```

See Also

elcsd.conf(5), eli(8), rc(8), uerf(8)
Guide to the Error Logger

Name

eli – error log initialization program

Syntax

eli [*options*]

Description

The `eli` command initializes error logging. This command enables error logging of hardware and system-related error packets, or error messages, from the kernel errorlog buffer. It can also disable error logging, reconfigure error logging parameters, or initialize the kernel errorlog buffer.

Options

–d	Disables error logging.
–e	Enables error logging in multiuser mode.
–f	Forces the subsequent option; the system will not prompt. This is the only `eli` command option you can use with another option.
–h	Prints information about the `eli` command.
–i	Initializes the kernel errorlog buffer. The previous contents of the errorlog buffer are lost.
–l	Logs a one-line status message to the kernel errorlog buffer.
–n	Only supported for local error logging. Disables logging error packets to disk by the `elcsd` daemon. High priority error messages continue to be printed at the console. Note that error log packets can be viewed by using the `uerf–n` option in real time, but are never written to the disk. For this reason, this option is rarely used.
–q	Suppresses the periodic display, on the console, of the missed error message that results from a full kernel errorlog buffer.
–r	Reconfigures error logging. Use this option after changing the `/etc/elcsd.conf` file.
–s	Enables error logging in single-user mode.
–w	Enables the missed error message to appear on the console every 15 minutes. This option is the opposite of `-q`.

Examples

This example logs a one-line message into the errorlog file.

```
eli -f -l "This is a test message"
```

This example logs a message, up to and including the first new line, from the file `myfile`:

```
eli -f -l < myfile > /dev/null
```

Restrictions

You must have superuser privileges to use the `eli` command.

Only the **–f** option can be used with other `eli` options. You must use the syntax shown in the examples above.

Diagnostics

eli: Request Aborted.
The requested action was aborted. The reason for the aborted command is included with the output.

Files

`/etc/elcsd.conf`

See Also

elcsd.conf(5), elcsd(8), uerf(8)
Guide to the Error Logger

ext_srvtab (8krb)

Name

ext_srvtab – extracts service key files.

Syntax

/var/dss/kerberos/bin/ext_srvtab [*–n*] [*hostname* ...]

Arguments

hostname	Name of the service key file.

Description

The `ext_srvtab` utility extracts service key files from the Kerberos database. When you invoke the utility, it prompts you to enter the master key string for the database. If the `-n` option is specified, `ext_srvtab` fetches the master key from the master key cache file, produced by `kstash(8krb)`.

For each *hostname* that you specify in the command line, `ext_srvtab` creates the service key file *hostname-new-srvtab*. This file contains all the entries in the Kerberos database with an instance field that matches *hostname*. In addition, the *hostname-new-srvtab* file contains all the keys registered for Kerberos services that can run on the host specified in *hostname*.

Options

–n	Causes `ext_srvtab` utility to fetch the master key from the master key cache file.

Files

```
/var/dss/kerberos/dbase/principal.pag
/var/dss/kerberos/dbase/principal.dir
/var/dss/kerberos/dbase/principal.ok
```

See Also

kdb_edit(8krb), kdb_init(8krb), kdb_util(8krb), kdb_destroy(8krb)

Name

fingerd – remote user information server

Syntax

/etc/fingerd

Description

The `fingerd` program is a protocol that provides an interface to the Name and Finger programs at several network sites. The program returns an in depth status report on either the system currently in use or on a particular user whether that user is logged in or not.

The `fingerd` program is never executed directly by a user, but rather by the ULTRIX Internet listener daemon `/etc/inetd`. When `/etc/inetd` is listening on port 79, it passes any requests that it receives to `/etc/fingerd`. Once `fingerd` receives control, it reads a single command line terminated by a `<CTRL/F>` which is passed to `finger`(1). The `fingerd` command closes its connections as soon as the output is finished.

If the line is null, for example a `<CTRL/F>` is sent, then `finger` returns a default report that lists all the users logged into the system at that moment.

If a user name is specified as `eric^F`, for example, more extended information is listed for only that particular user, whether he is logged in or not. Allowable names in the command line include both login names and user names. If a name is ambiguous, however, all possible derivations are returned.

See Also

finger(1), services(5), inetd(8c)

Name

fitset – determine if subset fits on a system

Syntax

fitset [**–d**] [*root-path*]

Description

The `fitset` utility is used to determine if the files in a software subset will fit on a system.

Subset inventory records are read from the standard input. For each record, the space required to install the file described in that record is deducted from the available free space on the file system to which it would be installed. Only currently mounted UFS file systems are used in the computations. Space requirements for files already on the disk will be modified to account for the size of the resident copy. After all records have been read, the free space computed for all file systems is checked. If the space required to install the files would cause any file system to be more than 90% full, `fitset` returns an exit status of –1.

The `setld` utility uses `fitset` to size all subsets before attempting to install them. The *root-path* argument is the pathname of the top directory for the hierarchy into which the files are going to be installed. If no *root-path* is specified, the directory '/' is assumed.

Options

–d Enable debugging. This will make `fitset` print voluminous status information on standard output. This information is the initial file system statistics, the file system location of the file from each input record and the statistics for the file system after the space required to install the file has been deducted.

Restrictions

NFS mounts are ignored. If software would be installed to an NFS mounted directory, it is sized against the file system containing the NFS mount point.

The program does not detect the use of symbolic links in paths to *root-path* or any of the mount points. This can cause `fitset` to size a subset incorrectly if *root-path* is a symbolic link or a symbolic link exists in the path of any of the pathnames used with the `mount` command to mount local file systems.

Examples

To determine if a particular subset will fit on the system, redirect the contents of the subset inventory file into `fitset`. For example:

```
fitset < /usr/etc/subsets/ULTUUCP400.inv
```

To determine if the same subset will fit in a hierarchy rooted at `/var/tmp/root`, the command would be:

```
fitset /var/tmp/root < /usr/etc/subsets/ULTUUCP400.inv
```

Diagnostics

fitset: root path must be absolute
A relative pathname was specified for *root-path.* This path must be absolute.

fitset: cannot stat *root-path (error message)*
The *root-path* cannot be accessed. The error message provides more information.

fitset: *root-path* **is not a directory.**
Either *root-path* is not a directory or it is a symbolic link to something which is not a directory.

Files

`/usr/etc/subsets/*.inv`
Subset inventory files

See Also

stl_inv(5), setld(8)
Guide to Preparing Software for Distribution on ULTRIX Systems

flcopy (8v)

Name

flcopy – copier for floppy

Syntax

/etc/flcopy [**–h**] [**–t***n*]

Description

The `flcopy` command copies the console floppy disk (opened as `/dev/floppy`) to a file created in the current directory, named `floppy`, then prints the message `Change Floppy, hit return when done.` Then `flcopy` copies the local file back out to the floppy disk.

The `-h` option to `flcopy` causes it to open a file named `floppy` in the current directory and copy it to `/dev/floppy`. The `-t` option causes only the first *n* tracks to participate in a copy.

Files

`/dev/floppy`	
`/dev/rrx??`	
`floppy`	In the current directory

See Also

cfl(4), rx(4), rxformat(8v)

Name

format – how to format disk packs

Description

There are two ways to format disk packs. The simplest is to use the `format` program. The alternative is to use the DEC standard formatting software which operates under the DEC diagnostic supervisor. This manual page describes the operation of `format,` then concludes with some remarks about using the DEC formatter.

The `format` program is a standalone program used to format and check disks prior to constructing file systems. In addition to the formatting operation, `format` records any bad sectors encountered according to DEC Standard 144. Formatting is performed one track at a time by writing the appropriate headers and a test pattern and then checking the sector by reading and verifying the pattern, using the controller's ECC for error detection. A sector is marked bad if an unrecoverable media error is detected, or if a correctable ECC error greater than 5 bits in length is detected (such errors are indicated as ''ECC'' in the summary printed upon completing the format operation). After the entire disk has been formatted and checked, the total number of errors are reported, any bad sectors and skip sectors are marked, and a bad sector forwarding table is written to the disk in the first five even numbered sectors of the last track. The `format` may be used on any UNIBUS or MASSBUS drive supported by the *up* and *hp* device drivers which uses 4-byte headers (everything except RP's).

The test pattern used during the media check may be selected from one of: 0xf00f (RH750 worst case), 0xec6d (media worst case), and 0xa5a5 (alternating 1's and 0's). Normally the media worst case pattern is used.

The `format` program also has an option to perform an extended ''severe burnin,'' which makes 46 passes using different patterns. Using this option, sectors with any errors of any size are marked bad. This test runs for many hours, depending on the disk and processor.

Each time `format` is run a completely new bad sector table is generated based on errors encountered while formatting. The device driver, however, will always attempt to read any existing bad sector table when the device is first opened. Thus, if a disk pack has never previously been formatted, or has been formatted with different sectoring, five error messages will be printed when the driver attempts to read the bad sector table; these Odiagnostics should be ignored.

Formatting a 400 megabyte disk on a MASSBUS disk controller usually takes about 20 minutes. Formatting on a UNIBUS disk controller takes significantly longer. For every hundredth cylinder formatted `format` prints a message indicating the current cylinder being formatted. (This message is just to reassure people that nothing is is amiss.)

The `format` program uses the standard notation of the standalone i/o library in identifying a drive to be formatted. A drive is specified as *zz*(*x*,*y*), where *zz* refers to the controller type (either *hp* or *up*), *x* is the unit number of the drive; 8 times the UNIBUS or MASSBUS adaptor number plus the MASSBUS drive number or UNIBUS drive unit number; and *y* is the file system partition on drive *x* (this should always be 0). For example, ''hp(1,0)'' indicates that drive 1 on MASSBUS adaptor 0 should be formatted; while ``up(10,0)'' indicates UNIBUS drive 2 on UNIBUS adaptor 1 should be formatted.

Before each formatting attempt, `format` prompts the user in case debugging should be enabled in the appropriate device driver. A carriage return disables debugging information.

The `format` should be used prior to building file systems (with `newfs(8)`) to insure all sectors with uncorrectable media errors are remapped. If a drive develops uncorrectable defects after formatting, the program `badsect(8)` must be used.

Examples

A sample run of `format` is shown below. In this example (using a VAX-11/780), `format` is loaded from the console floppy; on an 11/750 `format` will be loaded from the root file system. Boldface means user input. As usual, a number sign (#) or an at sign (@) can be used to edit input.

```
>>>L FORMAT
        LOAD DONE, 00004400 BYTES LOADED
>>>S 2
Disk format/check utility

Enable debugging (0=none, 1=bse, 2=ecc, 3=bse+ecc)? 0
Device to format? hp(8,0)
(error messages may occur as old bad sector table is read)
Formatting drive hp0 on adaptor 1: verify (yes/no)? yes
Device data: #cylinders=842, #tracks=20, #sectors=48
Available test patterns are:
        1 - (f00f) rh750 worst case
        2 - (ec6d) media worst case
        3 - (a5a5) alternating 1's and 0's
        4 - (ffff) Severe burnin (takes several hours)
Pattern (one of the above, other to restart)? 2
Start formatting...make sure the drive is on line
 ...
(soft ecc's and other errors are reported as they occur)
 ...
(if 4 write check errors were found, the program terminates like this...)
 ...
Errors:
Write check: 4
Bad sector: 0
ECC: 0
Skip sector: 0
Total of 4 hard errors found.
Writing bad sector table at block 808271
(808271 is the block # of the first block in the bad sector table)
Done
(...program restarts to allow formatting other disks)
(...to abort halt machine with ^P)
```

Diagnostics

The diagnostics are intended to be self explanatory.

Using DEC Software To Format

Caution

These instructions are for people with 11/780 CPU's." The steps needed for 11/750 or 11/730 CPUs are similar, but not covered in detail here.

The formatting procedures are different for each type of disk. Listed here are the formatting procedures for RK07's, RP0X, and RM0X disks.

You should shut down ULTRIX and halt the machine to do any disk formatting. Make certain you put in the pack you want formatted. It is also a good idea to spin down or write protect the disks you don't want to format, just in case.

Formatting an RK07

Load the console floppy labeled, "RX11 VAX DSK LD DEV #1" in the console disk drive, and type the following commands:

```
>>>BOOT
DIAGNOSTIC SUPERVISOR.  ZZ-ESSAA-X5.0-119  23-JAN-1980 12:44:40.03
DS>ATTACH DW780 SBI DW0 3 5
DS>ATTACH RK611 DMA
DS>ATTACH RK07 DW0 DMA0
DS>SELECT DMA0
DS>LOAD EVRAC
DS>START/SEC:PACKINIT
```

Formatting an RP0X

Follow the above procedures except that the ATTACH and SELECT lines should read:

```
DS>ATTACH RH780 SBI RH0 8 5
DS>ATTACH RP0X RH0 DBA0          (RP0X is, e.g. RP06)
DS>SELECT DBA0
```

This is for drive 0 on mba0; use 9 instead of 8 for mba1, etc.

Formatting an RM0X

Follow the above procedures except that the ATTACH and SELECT lines should read:

```
DS>ATTACH RH780 SBI RH0 8 5
DS>ATTACH RM0X RH0 DRA0
DS>SELECT DRA0
```

Do not forget to put your ULTRIX console floppy back in the floppy disk drive.

See Also

bad144(8), badsect(8), newfs(8)

Name

fsck – check and repair file system

Syntax

/etc/fsck [**-p -P**] [*filesystem* ...]
/etc/fsck [**-b** *block*] [**-y**] [**-n**] [*filesystem*] ...

Description

The `fsck` command checks and corrects either a standard set of file systems or the specified file systems for consistency. This command is normally used in the script `/etc/rc` during automatic reboot. In this case, `fsck` reads the `/etc/fstab` file to determine which UFS file systems to check. It uses the `fstab` information to inspect groups of disks in parallel, taking advantage of I/O overlap to check the file systems as quickly as possible.

The `fsck` command makes a number of passes to check the file systems for consistency. Usually, the root file system is checked on pass 1, other root file systems such as partition `a` are checked on pass 2, and other small file systems are checked on separate passes. For example, the `d` file systems are usually checked on pass 3 and the `e` file systems are usually checked on pass 4. The large user file systems are usually checked on the final pass. A pass number of 0 in `/etc/fstab` causes a disk to not be checked. Similarly, partitions that are not shown to be mounted with `rw` or `ro` are not checked.

The `-p` option should be used to check file systems. The generic file system interface, `gfs`, causes `fsck` to realize when a file system is unmounted cleanly and thus prevents `fsck` from doing the check. File systems are unmounted cleanly only when an error-free shutdown has been performed or the file system was unmounted. However, a timeout factor is used by `fsck` to determine if `fsck` should be run regardless of the value of the clean byte. The timeout factor is initially set to 20 and is decremented when any one of three events occur:

- A file system is mounted,
- 10,000 updates have occurred
- A file system was updated and `fsck` occurred more than 60 days prior

When the timeout factor reaches 0, `fsck` will automatically check it. This factor can be changed with `tunefs`. If the `-P` option is used, the parallel consistency checks are performed like the `-p` option regardless of how the file system was unmounted.

If an attempt is made to check a mounted file system using the block device, `fsck` will report *filesystem:* `NO WRITE ACCESS` and will check the filesystem as if the `-n` option is selected.

The system ensures that only a restricted class of file system inconsistencies can occur unless hardware or software failures intervene. The inconsistencies are limited to:

Unreferenced inodes

Link counts in inodes are too large

Missing blocks in the free list

Blocks in the free list are also in files

Counts in the superblock are wrong

These are the only inconsistencies that `fsck` corrects with either the `-p` or `-P` option. If `fsck` encounters other inconsistencies, it exits with an abnormal return status and an automatic reboot will then fail. For each corrected inconsistency one or more lines are printed identifying the file system on which the correction will take place and the nature of the correction. If any inconsistencies occur, the message `**** FILE SYSTEM MODIFIED, VERIFYING` is printed and `fsck` runs again to verify that the appropriate changes were made. After correcting a file system, `fsck` prints the number of files on that file system and the number of used and free blocks and also the percent of fragments vs blocks. When the fragmentation exceeds 5% it is recommended that the file system be dumped to tape, `newfs,` and restored. Also, a clean byte is set for the checked file system. The root file system is checked regardless of whether the clean byte is set.

Without the `-p` or `-P` options, `fsck` audits and interactively repairs inconsistent conditions for file systems. If the file system is inconsistent, the operator is prompted before each correction is attempted. It should be noted that a number of the corrective actions which are not fixable using the `-p` or `-P` options will result in some loss of data. The amount and severity of data lost may be determined from the diagnostic output. The default action for each consistency correction is to wait for the operator to respond `yes` or `no`. If the operator does not have write permission, `fsck` defaults to a `-n` action.

If no file systems are given to `fsck`, then a default list of file systems is read from the file `/etc/fstab`. The `fsck` command only checks file systems of type UFS.

The `fsck` command checks for the following inconsistencies:

Blocks claimed by more than one inode or the free list.

Blocks claimed by an inode or the free list outside the range of the file system.

Incorrect link counts.

Size checks; directory size not of proper format.

Bad inode format.

Blocks not accounted for.

Directory checks; file pointing to unallocated inode; inode number out of range.

Superblock checks; more blocks for inodes than there are in the file system.

Bad free block list format.

Total free block or free inode count incorrect.

If `fsck` detects allocated but unreferenced files and directories, it prompts you before placing them in the `lost+found` directory. The only restriction is that the directory `lost+found` must exist in the root of the file system being checked and must have empty slots before `fsck` is run. If necessary, the `lost+found` directory can be enlarged by creating many files in the directory and then removing them.

General users can run `fsck` on file systems with certain restrictions. The user must have execute permissions on the device and general users cannot run `fsck` on a mounted file system.

Options

–b	Use the block specified immediately after the flag as the superblock for the file system. Block 32 is always an alternate superblock.
–y	Assume a `yes` response to all questions asked by `fsck`; this should be used with caution as this allows `fsck` to continue after essentially unlimited trouble has been encountered.
–n	Assume a `no` response to all questions asked by `fsck`; do not open the file system for writing.
–p	Check a file system that was not unmounted cleanly.
–P	Check a file system regardless of how it was unmounted.

Restrictions

Inode numbers for `.` and `..` in each directory should be checked for validity. The `fsck` command will not allow checking a raw device if the block device is mounted.

Files

`/etc/fstab`	Contains default list of file systems to check

See Also

getmnt(2), fstab(5), ufs(5), crash(8v), mkfs(8), mklost+found(8), mount(8), mount(8ufs), newfs(8), reboot(8), tunefs(8)

Name

fsirand – install random inode generation numbers

Syntax

fsirand [**–p**] *special*

Description

The `fsirand` command installs random inode generation numbers on all the inodes on device *special*. This helps increase the security of file systems exported by NFS.

The `fsirand` command must be used only on an unmounted file system that has been checked with `fsck`(8). The only exception is that it can be used on the root file system in single-user mode, if the system is then immediately rebooted.

Options

–p	Print out the generation numbers for all the inodes, but do not change the generation numbers.

Name

fsx – file system exerciser

Syntax

/usr/field/fsx [**–h**] [**–o***file*] [**–t***n*] [**–f***path*] [**–p***m*]

Description

The `fsx` exerciser exercises a file system by spawning up to 250 (the default is 20) processes that create, open, write, close, open, read, validate, close, and unlink a test file. These test files are created in `/usr/field` (the default) unless the **–f***path* option is used. The exerciser will run until <CTRL/C> or **kill -15** *pid* is sent to the process.

A logfile is made in `/usr/field` for you to examine and then remove. If there are errors in the logfile, make sure you check the `/usr/adm/syserr/syserr.<hostname>` file, because that is where the driver and kernel error messages are saved.

Options

The `fsx` options are:

–h	Print the help messages for the `fsx` command.
–o*file*	Save the output diagnostics in *file*.
–t*n*	Run time in minutes (*n*). The default is to run until the process receives a <CTRL/C> or a **kill -15** *pid*.
–p*m*	Number (*m*) of `fsx` processes to spawn. The maximum is 250; the default is 20.
–f*path*	Path name of directory on file system you wish to test. For example, `/mnt` or `/usr`. The default is `/usr/field`.

Examples

The following example runs 10 `fsx` processes on `/mnt` until the process receives a <CTRL/C> or **kill -15** *pid*:

```
% /usr/field/fsx -p10 -f/mnt
```

The following example runs 20 `fsx` processes on `/usr/field` for 120 minutes in the background:

```
% /usr/field/fsx -t120 &
```

Restrictions

If there is a need to run a system exerciser over an NFS link or on a diskless system there are some restrictions. For exercisers that need to write into a file system, such as `fsx`(8), the target file system must be writable by root. Also the directory, in which any of the exercisers are executed, must be writable by root because temporary files are written into the current directory. These latter restrictions are sometimes difficult to overcome because often NFS file systems are mounted in a way that prevents root from writing into them. Some of the restrictions may be overcome by copying the exerciser to another directory and then executing it. Avoid using the `fsx` exerciser over an NFS or diskless file system.

See Also

Guide to System Exercisers

ftpd(8c)

Name

ftpd – DARPA Internet File Transfer Protocol server

Syntax

/usr/etc/ftpd [**–d**] [**–l**] [–ttimeout]

Description

The `ftpd` server is the DARPA Internet File Transfer Protocol server process. The server uses the TCP protocol and is invoked by `inetd(8c)` when it receives a connection on the port specified in the `ftp` service specification. For further information, see `services(5)`.

The `ftp` server currently supports the following `ftp` requests. Case is not distinguished.

Request	**Description**
ABOR	Abort previous command
ACCT	Specify account
ALLO	Allocate storage
APPE	Append to a file
CDUP	Change to parent of current working directory
CWD	Change working directory
DELE	Delete a file
HELP	Give help information
LIST	Give list of files in a directory (`ls -lg`)
MKD	Make a directory
MODE	Specify data transfer *mode*
NLST	Give name list of files in directory (`ls`)
NOOP	Do nothing
PASS	Specify password
PASV	Prepare for server-to-server transfer
PORT	Specify data connection port
PWD	Print the current working directory
QUIT	Terminate session
RETR	Retrieve a file
RMD	Remove a directory
RNFR	Specify rename-from file name
RNTO	Specify rename-to file name
STOR	Store a file

STOU	Store a file with a unique name
STRU	Specify data transfer *structure*
TYPE	Specify data transfer *type*
USER	Specify user name
XCUP	Change to parent of current working directory
XCWD	Change working directory
XMKD	Make a directory
XPWD	Print the current working directory
XRMD	Remove a directory

The remaining `ftp` requests specified in Internet RFC 959 are recognized, but not implemented.

The `ftpd` server interprets file names according to the globbing conventions used by `csh(1)`. This allows users to utilize the metacharacters *?[]{}~.

The `ftpd` server authenticates users according to three rules:

1. The user name must be in the password database, `/etc/passwd`, and not have a null password. In this case a password must be provided by the client before any file operations may be performed.

2. The user name must not appear in the file `/etc/ftpusers`.

3. If the user name is anonymous or `ftp`, an anonymous `ftp` account must be present in the password file (user `ftp`). In this case the user is allowed to log in by specifying any password (by convention this is given as the client host's name).

In the last case, `ftpd` takes special measures to restrict the client's access privileges. The server performs a `chroot(2)` command to the home directory of the `ftp` user. To prevent system security from being breached, it is recommended that the `ftp` subtree be constructed with care. Thus the following rules are recommended:

~ftp)	Make the home directory owned by `ftp` and unwritable by anyone.
~ftp/bin)	Make this directory owned by the superuser and unwritable by anyone. The program `ls(1)` must be present to support the list commands. This program should have mode 111.
~ftp/etc)	Make this directory owned by the superuser and unwritable by anyone. The files `passwd`(5) and `group`(5) must be present for the `ls` command to work properly. These files should be mode 444.
~ftp/pub)	Make this directory mode 777 and owned by `ftp`. Place the files, which are to be accessible by the anonymous account, in this directory.

Options

–d	Enables certain debugging messages that are printed by ftpd.
–l	Logs each ftp session to the `syslog`.
–t	Sends the inactivity timeout period to *timeout;* otherwise, the `ftp` server will timeout an inactive session after 15 minutes.

Restrictions

Support does not exist for aborting commands.

The use of an anonymous account is inherently dangerous and should be avoided when possible.

The server must run as the superuser to create sockets with privileged port numbers. The server maintains an effective user id of the logged in user, reverting to the superuser only when binding addresses to sockets. The possible security holes have been extensively scrutinized, but are possibly incomplete.

Files

`/etc/ftpusers`
Contains the list of unauthorized users

See Also

ftp(1c), services(5), inetd(8c), syslog(8)

Name

fverify – verify software subsets

Syntax

fverify [–yn]

Description

The `fverify` command reads subset inventory records from standard input and verifies that the attributes for the files on the system match the attributes listed in the corresponding records. Invoking `fverify` with no arguments causes it to report errors to the standard output and prompt the user for confirmation before making corrections.

Missing files and inconsistencies in file size, checksum, user id, group id, mode, and file type are reported. If inconsistencies in user id, group id or mode are detected the user is asked if they are to be corrected. If a missing file is a directory, it will be created. All errors and informational messages are logged to the file `/var/adm/fverifylog`.

The `setld` utility uses `fverify` when installing subsets to guarantee correct installation.

Options

Specifying both options will cause the **–y** option to be ignored.

–n Error reporting only. When specified, `fverify` reports errors and does not perform any fixes.

–y Fixes only. When specified, `fverify` fixes modes, UIDs and GIDs on any files which have incorrect values for these attributes. No user input is required.

Restrictions

Problems with file size, file type, and file checksum cannot be corrected.

Because the subset inventories give file names as relative paths, run `fverify` in the root directory to which the software is installed.

Many of the files on the system cannot be read or modified without appropriate privilege. Attempting to run `fverify` without appropriate privilege may result in an excess of access errors being reported.

Examples

To use `fverify` to report verification problems with the BASE subset on ULTRIX, Version 4.0 (VAX), use the following command:

```
% cd /
% /etc/stl/fverify -n </usr/etc/subsets/ULTBASE400.inv
```

To correct all problems with the same subset without being presented any prompts or diagnostic output, type:

```
% cd /
% /etc/stl/fverify -y < /usr/etc/subsets/ULTBASE400.inv
```

To interactively repair verification problems with the COMM subset on ULTRIX, Version 4.0 (RISC), use the fverify command with no arguments:

```
% cd /
% /etc/stl/fverify < /usr/etc/subsets/UDTCOMM400.inv
```

Diagnostics

fverify: out of memory
Not enough memory is available from the system.

pathname: **cannot stat** (*error-message*)
The file *pathname* is listed in the inventory but is not on the system. The *error-message* explains why.

pathname: **file type '*x*' should be '*y*'**
The file is listed in the inventory as being of type '*y*', but the copy of the file on the disk is type '*x*'. The file type codes are:

b	block device files
c	character device files
d	directory files
f	regular files
l	hard links
p	fifos (named pipes)
s	symbolic links

pathname: **checksum** *n* **should be** *m*
There is a checksum error on *pathname*. The values *n* and *m* are the actual and expected values.

pathname: **size** *n* **should be** *m*
The size of file *pathname* is not as expected. The numbers *n* and *m* are the actual and expected values.

pathname: **gid** *n* **should be** *m*
The group id for *pathname* is not as expected. The group ids *n* and *m* are the actual and expected values.

pathname: **uid** *n* **should be** *m*
The user id for *pathname* is not as expected. The user ids *n* and *m* are the actual and expected values.

pathname: **permissions** *string1* **should be** *string2*
The permissions for *pathname* are not as expected. *String1* and *string2* are the actual and expected permissions. The format of *string1* and *string2* is the same as that used by the `ls` command.

Creating directory *pathname*
The directory *pathname* is listed in the input inventory but does not exist on the system. The `fverify` command will attempt to create the directory.

cannot correct *pathname* (*error-message*)
A problem reported with *pathname* could not be corrected. The *error-message* explains why.

pathname **corrected**
A problem with *pathname* was successfully corrected.

Cannot create dir *pathname* (*error-message*)
The `fverify` program could not create a directory it was attempting to create. The accompanying *error-message* explains why.

n **verification errors encountered.**
m **corrections performed.**
Before exiting, the `fverify` program prints these statistics describing what had been done.

The exit status from `fverify` is the total number of errors detected minus the total number of successful fixes.

Files

`/usr/etc/subsets/*.inv`	Subset inventory files
`/var/adm/fverifylog`	Log File

See Also

ls(1), stl_inv(5), setld(8)
Guide to Preparing Software for Distribution on ULTRIX Systems

genra(8)

Name

genra – produce distribution RA60 media

Syntax

/etc/genra [-{wv}] [*hostname*:]*product_code* **special**

Description

The `genra` utility is used to produce RA60 distribution media. The utility searches `/etc/kitcap` for the *product_code* (as provided by the user on the command line), creates a new file system on the partition defined in the kit description, mounts the disk, transfers files and subsets as described in the kitcap kit description, uses `sum`(1) to verify files for accuracy in transfer, and then unmounts the file system. The `genra` utility can create multiple directories on the RA60 for separate products, and can also combine multiple products into one product, all controlled by `/etc/kitcap`.

The optional *hostname:* argument is the name of a remote TCP/IP network machine that contains the kitcap file. The utility will search `/etc/kitcap` on the remote machine for the *product_code* and use it for creating the media. The colon (:) is a required delimiter for TCP/IP networks, and there is no space between the colon and the *product_code*. For example, if the product code was ULT-2.2-V-BW, and the kitcap file to be used was on node "mynode", the proper syntax for this option would be

```
mynode:ULT-2.2-V-BW
```

The *product_code* is a user defined code that is located in `/etc/kitcap`, and describes the partition, directories, and subsets, that make up a kitcap description. It can be any set of numbers or letters, and is usually formed in a way to have some meaning as to the product that it describes. For example, if the product name is MYPRODUCT and it is version 1, a proper *product_code* for that product might be MYP010.

The **special** argument indicates that the device is a special device such as `/dev/rra1a`. The actual partition that the utility will write the files on is defined in the kitcap description for the *product_code* that is being used. Therefore, it makes no difference what partition is appended to the special device file used on the command line.

Options

–w Write only.

–v Verify only.

If neither option appears on the command line, then by default, the utility will write, then verify, the files in the kit descriptor.

Restrictions

You must be a superuser to run this program. If the optional hostname is used, you must be able to access files on the remote host as superuser.

Files

`/dev/ra?a`	ra device special files
`/etc/kitcap`	Kit descriptor database

See Also

gentapes(1), sum(1), kitcap(5)

Name

gentapes – produce distribution mag tape media

Syntax

/etc/gentapes [-{wv}] [*hostname*:]*product_code* **special**

Description

The `gentapes` utility is used to produce MT9 or TK50 mag tape distribution media. The utility searches `/etc/kitcap` for the *product_code* as provided by the user on the command line, plus the letters "TK" which is appended to the product code by the utility. If a corresponding kit descriptor is found in `/etc/kitcap`, the utility uses the information in the kit descriptor to copy files to the tape media mounted on tape drive special. Once all files have been transferred, the tape is rewound and each file is verified by using the results of `sum` and comparing it to sums given in the kit image file. The `gentapes` utility can support multi-volume tape kits, and multiproduct tape kits. See `kitcap`(5) for information on how to set up these features.

The optional *hostname:* is the name of a remote TCP/IP network machine that contains the kitcap file. The utility will search `/etc/kitcap` on the remote machine for the *product_code* and use it for creating the media. The colon (:) is a required delimiter for TCP/IP networks, and there is no space between the colon and the *product_code*. For example, if the product code was ULT-2.2-V-BW, and the kitcap file to be used was on node "mynode", the proper syntax for this option would be

```
mynode:ULT-2.2-V-BW
```

The *product_code* is a user defined code that is located in `/etc/kitcap`, and describes the partition, directories, and subsets, that make up a kitcap description. It can be any set of numbers or letters, and is usually formed in a way to have some meaning as to the product that it describes. For example, if the product name is MYPRODUCT and it is version 1, a proper *product_code* for that product might be MYP010.

The **special** argument is a special device such as /dev/nrmt?l.

Options

–w Write only.

–v Verify only.

If neither option appears on the command line, then by default, the utility will write, then verify, the files in the kit descriptor.

Restrictions

You must be a superuser to run this program. If the optional hostname is used, you must be able to access files on the remote host as superuser.

Files

`/dev/nrmt?l`	ra device special files
`/etc/kitcap`	Kit descriptor database

See Also

genra(1), sum(1), kitcap(5)

Name

getauth, setauth, rmauth – auth database maintenance

Syntax

getauth [*username*]

setauth

rmauth *username*

Description

The command `getauth` takes as it's only argument a user name or UID. If a user name is supplied it is converted to a UID by searching through `/etc/passwd`. The UID is then used to look up the users entry in the `/etc/auth` database. If an entry is found it is converted to an ASCII string with a syntax resembling that of the passwd file and printed out as a single line. If no entry is found nothing is printed and an exit status of '1' is returned.

```
# getauth username
1000:idvidfy8d:1920129:3600:2678400:0e:0:1000:0:00:00
```

The first field is the UID of the entry which is used as the key into the database. Then follows: the encrypted password, password modification time, minimum password lifetime, maximum password lifetime, account mask, login failure count, audit ID, audit control, audit mask, and a reserved field.

If the optional username argument is not supplied to getauth it will produce an output line for every entry in the auth database.

The `setauth` command expects one or more lines from the standard input which must be of a form identical to that produced by the `getauth` command. The `setauth` command converts and stores these lines into the `auth` database, one entry per line, replacing any entry already existing for the given UID. By piping the output of the `getauth` command into the input of the `setauth` command an expensive NOP can be produced:

```
# /usr/etc/sec/getauth | /usr/etc/sec/setauth
```

The `rmauth` command expects exactly one argument, the user name or UID of an auth entry to be deleted. If the entry is found it is erased and deleted. If it is not found no action is taken and an exit status of 1 is returned.

Restrictions

Only the superuser and members of the group `authread` may read information from the auth database. Only the superuser may modify the auth database.

Diagnostics

An exit value of 0 indicates a successful operation. An exit status of 1 indicates the entry was not found on a lookup or deletion operation. Any other exit status indicates an error.

Files

```
/etc/auth.[pag,dir]
/etc/passwd
```

See Also

getauthuid(3), getpwent(3), auth(5), edauth(8)
Security Guide for Administrators

getnode(8)

Name

getnode – display one or more entries from the nodes database

Syntax

/etc/getnode [*node...*]

Description

For each given *node* argument, `getnode` displays the corresponding node entry (or entries) from the nodes database. The nodes database is the one used by DECnet. If you do not specify any *node* argument(s), all of the entries in the nodes database are displayed.

The *node* is either the node address or the node name for each node entry that you want to display. (Note that you can specify more than one *node* argument in a single command.)

A node address is a decimal integer in the range of 1 to 1023 for single area networks, or has the format *a.n* for multiarea networks, where *a* is the network area number (a decimal integer in the range of 2 to 63) and *n* is the node number (a decimal integer in the range of 1 to 1023).

A node name can be from 1 to 6 alphanumeric characters, including at least 1 alphabetic character.

Examples

```
# /etc/getnode 44.70 mynode <RET>
```

This command displays the entries in the nodes database for nodes 44.70 and mynode.

```
# /etc/getnode <RET>
```

This command displays all of the entries in the nodes database.

```
# /etc/getnode lttwi <RET>
```

This command displays the entry in the nodes database for node lttwi.

See Also

addnode(8), ccr(8), load(8), mop_mom(8), remnode(8), trigger(8)
Guide to Ethernet Communication Servers

Name

gettable – get NIC format host tables from a host

Syntax

/etc/gettable *host*

Description

The `gettable` program is used to obtain the NIC standard host tables from a ''nicname'' server. The indicated *host* is queried for the tables. The tables, if retrieved, are placed in the file *hosts.txt*.

The `gettable` program operates by opening a TCP connection to the port indicated in the service specification for ''nicname''. A request is then made for ''ALL'' names and the resultant information is placed in the output file.

The `gettable` program is best used in conjunction with the `htable(8)` program which converts the NIC standard file format to that used by the network library lookup routines.

See Also

intro(3n), htable(8)

Name

getty – set terminal mode

Syntax

/etc/getty [*type*] [*tty*]

Description

The `getty` routine is one of several (`init, getty, login, shell`) by which users gain access to the ULTRIX system from a terminal. The `getty` routine initializes a terminal line, reads a login name, and invokes `login`(1). While reading the name, the `getty` routine tries to adapt the system to the speed and type of terminal on the line specified by the `tty` argument.

The `init` command typically invokes `getty`, as directed by the *command* field in the `/etc/ttys` file.

The `getty` routine first tries to initialize the line. It examines `/etc/ttys` and sets up the line for local or remote connections, as appropriate. Next, the `getty` routine calls `vhangup`(2) to revoke access to the terminal by any background processes that could have `/dev/tty` open. The `getty` routine then opens `/dev/tty` for reading and writing. File descriptors 0, 1, and 2 become the standard input, output, and diagnostic devices. If the terminal line is connected to a modem, the open is not completed until someone dials up and establishes carrier on the channel.

If a terminal exists, but an error occurs when trying to open the terminal, the `getty` routine writes a message to the system console. The message is repeated every 10 minutes until the terminal is available, or the `/etc/ttys` entry for the terminal is modified to indicate that the terminal is off, and `init`(8) is notified by a hangup.

The `getty` routine checks every minute to see if the terminal is still off.

Next, the `getty` routine reads a login name, terminated by a newline or carriage-return character. For a carriage return, the system is set to treat carriage returns appropriately. For further information, see `tty`(4).

The `getty` routine scans the user's name to see if it contains any lowercase alphabetic characters. If it does not, and the name is nonempty, the system is told to map any future uppercase characters into the corresponding lowercase characters.

Finally, `getty` calls `login` with the user's name as an argument.

Most of the default actions of `getty` can be changed with a suitable `gettytab` table.

The `getty` routine can be set to timeout after some interval with the *to* variable in the `gettytab`(5) table. Thus, if the user does not enter a login name after a reasonable amount of time, `getty` hangs up the dialup line.

Arguments

tty	The special device file in the `/dev` directory to open for the terminal (for example, ttyh0). If there is no argument or the argument is "+", the terminal line is assumed to be open as file descriptor 0.
type	Used to make `getty` treat the line specially. This argument is used as an

index into the gettytab(5) database, to determine the characteristics of the line. If there is no argument, or there is no such table, the default entry in gettytab(5) is used. If there is no /etc/gettytab, a set of system defaults is used.

The gettytab(5) entry is used to define specific terminal hardware attributes such as the baud rate and number of bits per character. Terminals which are setup to transmit and receive 8-bit characters must specify a gettytab entry which appropriately sets up the line. For example a terminal line setup to operate at 9600 baud with 8-bit characters may use the "8bit.9600" gettytab entry. If a terminal is setup to use 8-bit characters, but uses a 7-bit gettytab entry (such as "std.9600") the output from the getty program may be corrupted. This output corruption appears as multinational characters being generated by getty or login programs. When these programs operate in 7-bit mode, the high order bit of the character is used as software provided parity. This parity generation causes conventional ASCII characters to be transformed into multinational characters. By using an 8-bit gettytab entry, characters will be transmitted without any software generated parity.

If indicated by the table located, getty will clear the terminal screen, print a banner heading, and prompt for a login name. Usually either the banner or the login prompt will include the system hostname. Then the user's name is read, a character at a time. If a null character is received, it is assumed to be the result of the user pushing the break (interrupt) key. The speed is usually then changed and the ''login:'' prompt is displayed again. a second break changes the speed again and redisplays the ''login:'' prompt. Successive break characters cycle through some standard set of speeds.

Diagnostics

The getty uses syslog(3) to produce diagnostic messages. Therefore, the syslog configuration file will determine where the messages are printed. For further information, see syslog(8).

getty: *tty*: **cannot open**
A terminal which is turned on in the ttys file cannot be opened. This is probably because the requisite lines are either not configured into the system or the associated device was not attached during boot-time system configuration. The syslog error logging level is LOG_ERR.

getty: *command, tty* **failing, open blocked**
The getty routine tried a non-blocking open of the terminal line and the open still blocked. This can only happen on devices that have not implemented O_NDELAY. For further information, see open(2). The getty routine tries to continue but the line may not be properly initialized. The syslog error logging level is LOG_ERR.

getty: *command, tty* **open failed, reason**
The getty routine tried a non-blocking open of the terminal line and the open failed. The *reason* is the explanation produced by the perror(3) routine for why the open failed. The getty routine tries to continue but the line may not be properly initialized. The syslog error logging level is LOG_ERR.

getty: in use line *tty*
The *Itty* is in use by some other process. The `getty` routine will not initialize a line that is already in use. The `getty` routine will block until the line is no longer in use. See `tty`(4) and `open`(2) for discussions on shared lines. The syslog error logging level is LOG_INFO.

getty could not set pgrp, *reason*
The `getty` routine tried and failed to initialize the process group of the terminal to process group 0. See `tty`(4) for a discussion of process groups. The *reason* is the explanation produced by `perror`(3) for the failure. The syslog error logging level is LOG_INFO.

Files

`/etc/gettytab`	Data base describing terminal lines

See Also

login(1), tty(4), gettytab(5), ttys(5), init(8), syslog(8)

Name

halt – stop the processor

Syntax

/etc/halt [**–n**] [**–q**] [**–y**]

Description

The `halt` command writes information to the disks and then returns the processor to the monitor. The machine does not reboot.

Options

-n	Prevents the sync before stopping.
-q	Causes a quick halt, no graceful shutdown is attempted.
-y	Allows you to halt the system from a dialup.

See Also

reboot(8), shutdown(8)

halt(8)

Name

halt – stop the processor

Syntax

/etc/halt [**–n**] [**–q**] [**–y**]

Description

The `halt` command writes out sandbagged information to the disks and then stops the processor. The machine does not reboot, even if the auto-reboot switch is set on the console.

Options

–n	Prevents the sync before stopping.
–q	Causes a quick halt, no graceful shutdown is attempted.
–y	Halts the system from a dialup.

Restrictions

It is very difficult to halt a VAX, as the machine wants to then reboot itself. A rather tight loop suffices.

See Also

reboot(8), shutdown(8)

Name

hesupd – Hesiod update daemon for modifying BIND/Hesiod passwords

Syntax

`/usr/etc/hesupd`

Description

The Hesiod update daemon, `/usr/etc/hesupd`, is a server that handles password change requests from the `passwd` command and is run only on the BIND/Hesiod primary server serving the password database. The daemon changes the password entries on the BIND/Hesiod primary server.

The primary server is identified by the host name alias, bindmaster, which must exist on the server's host entry in the hosts database. The bindsetup command adds the host name alias, bindmaster to the `/var/dss/namedb/src/hosts` file if it does not already exist.

This daemon is not run by default, nor can it be started up from the `inetd` daemon. If you want to enable remote password updating for BIND/Hesiod, put an entry for `hesupd` in the `/etc/rc.local` file of the host serving as the primary server for the BIND/Hesiod `passwd` file. You can add the startup lines for `hesupd` to `/etc/rc.local` by running the `bindsetup` command.

Hesiod keeps a log file, `/var/adm/hesupd.log`, which records successful and unsuccessful password changes by uid.

Examples

This following example shows lines you can add to `/etc/rc.local` in order to start the Hesiod update daemon at boot time.

```
[ -f /usr/etc/hesupd ] && {
   /usr/etc/hesupd; echo -n ' hesupd' >/dev/console
}
```

Files

`/etc/rc.local`	Startup commands pertinent to a specific system
`/var/adm/hesupd.log`	Log of password changes

Default BIND Files:

`/var/dss/namedb/src/passwd`	BIND/Hesiod passwd file
`/var/dss/namedb/src/hosts`	BIND hosts file

See Also

bindsetup(8), passwd(1)
Guide to the BIND/Hesiod Service

htable (8)

Name

htable – convert NIC standard format host tables

Syntax

/etc/**htable** *file*

Description

The `htable` program is used to convert host files in the format specified in Internet RFC 810 to the format used by the network library routines. Three files are created as a result of running `htable`: *hosts*, *networks*, and *gateways*. The *hosts* file is used by the `gethostent(3n)` routines in mapping host names to addresses. The *networks* file is used by the `getnetent(3n)` routines in mapping network names to numbers. The *gateways* file is used by the routing daemon in identifying passive Internet gateways; see `routed(8c)` for an explanation.

If any of the files *localhosts*, *localnetworks*, or *localgateways* are present in the current directory, the file's contents is prepended to the output file without interpretation. This allows sites to maintain local aliases and entries which are not normally present in the master database.

The `htable` program is best used in conjunction with the `gettable(8c)` program which retrieves the NIC database from a host.

Restrictions

Does not properly calculate the *gateways* file.

See Also

intro(3n), gettable(8c)

Name

icheck – check inode consistency

Syntax

/etc/icheck [**–s**] [**–b** *numbers*] [*filesystem*]

Description

The `icheck` command is obsoleted for normal consistency checking by `fsck`(8).

The `icheck` command examines a file system, builds a bit map of used blocks, and compares this bit map against the free list maintained on the file system. If the file system is not specified, a set of default file systems is checked. The normal output of `icheck` includes a report of:

- The total number of files and the numbers of regular, directory, block special and character special files.
- The total number of blocks in use and the numbers of single-, double-, and triple-indirect blocks and directory blocks.
- The number of free blocks.
- The number of blocks missing, that is, not in any file nor in the free list.

The `-s` option causes `icheck` to ignore the actual free list and reconstruct a new one by rewriting the super-block of the file system. The file system should be dismounted while this is done; if this is not possible (for example if the root file system has to be salvaged) care should be taken that the system is quiescent and that it is rebooted immediately afterwards so that the old, bad in-core copy of the super-block will not continue to be used. Notice also that the words in the super-block which indicate the size of the free list and of the i-list are believed. If the super-block has been curdled these words will have to be patched. The `-s` option causes the normal output reports to be suppressed.

Following the `-b` option is a list of block numbers; whenever any of the named blocks turns up in a file, a diagnostic is produced.

The `icheck` command is faster if the raw version of the special file is used, since it reads the i-list many blocks at a time.

Diagnostics

For duplicate blocks and bad blocks (which lie outside the file system) `icheck` announces the difficulty, the i-number, and the kind of block involved. If a read error is encountered, the block number of the bad block is printed and `icheck` considers it to contain 0. 'Bad freeblock' means that a block number outside the available space was encountered in the free list. '*n* dups in free' means that *n* blocks were found in the free list which duplicate blocks either in some file or in the earlier part of the free list.

Restrictions

Since `icheck` is inherently two-pass in nature, extraneous diagnostics may be produced if applied to active file systems.

It believes even preposterous super-blocks and consequently can get core images.

Files

Default file systems vary with installation.

See Also

fs(5), clri(8), dcheck(8), fsck(8), ncheck(8)

Name

ifconfig – configure network interface parameter

Syntax

/etc/ifconfig *interface* [*address* [*dest_address*]] [*parameters*]

Description

The `ifconfig` command assigns an address to a network interface and/or configures network interface parameters. You must use `ifconfig` at boot time to define the network address of each interface present on a machine. You can also use it at a later time to redefine an interface's address. The *interface* parameter is a string of the form: name, unit, for example, en0. The address is either a host name present in the host name data base, `hosts`(5), or a DARPA Internet address expressed in the Internet standard dot notation.

Arguments

You can set the following parameters with `ifconfig`:

up	Marks an interface up.
down	Marks an interface down. When an interface is marked down, the system does not attempt to transmit messages through that interface.
trailers	Enables the use of a trailer link level encapsulation when sending. This is the default. If a network interface supports *trailers*, the system, when possible, encapsulates outgoing messages in a manner that minimizes the number of memory-to-memory copy operations performed by the receiver.
–trailers	Disables the use of a trailer link level encapsulation.
promisc	Enables the use of the `packetfilter`(4) in the promiscuous mode. The promiscuous mode allows the network interface to receive all the packets off the wire and pass it onto to the packet filter.
–promisc	Disables the promiscuous mode of the packet filter. This is the default.
arp	Enables the use of the Address Resolution Protocol in mapping between network level addresses and link level addresses. This is the default. This is currently implemented for mapping between DARPA Internet addresses and 10Mb/s Ethernet addresses.
–arp	Disables the use of the Address Resolution Protocol.
debug	Enables driver-dependent debugging code. Usually, this turns on extra console error logging.
–debug	Disables driver dependent debugging code.
netmask	Specifies how many bits of the address you wish to reserve for subdividing Class A and B networks into sub-networks. (Inet only).
dstaddr	Specifies the correspondent on the other end of a point to point link.
broadcast	Specifies the address you wish to use to represent broadcasts to the network.

ifconfig (8c)

The `ifconfig` command displays the current configuration for a network interface when no optional parameters are supplied.

Only the superuser can modify the configuration of a network interface.

Diagnostics

The `ifconfig` command returns messages indicating the specified interface does not exist, the requested address is unknown, the user tried to alter an interface's configuration but is not privileged.

See Also

netstat(1), intro(4n), packetfilter(4), MAKEDEV(8), pfconfig(8c), pfstat(8), rc(8)

Name

implog – IMP log interpreter

Syntax

/etc/implog [*options*]

Description

The `implog` program interprets the message log produced by `implogd`(8c).

If no arguments are specified, `implog` interprets and prints every message present in the message file.

Options

Options may be specified to force the printing of only a subset of the logged messages.

-c In addition to printing any data messages logged, show the contents of the data in hexadecimal bytes.

-D Do not show data messages.

-f Follow the logging process in action. This flag causes `implog` to print the current contents of the log file, then check for new logged messages every 5 seconds.

–h *host#* Show only those messages received from the specified host. (Usually specified in conjunction with an `imp`.)

–i *imp#* Show only those messages received from the specified `imp`.

–l [*link#*] Show only those messages received on the specified link. If no value is given for the link, the link number of the IP protocol is assumed.

–r Print the raw `imp` leader, showing all fields, in addition to the formatted interpretation according to type.

–t *message-type*
Show only those messages received of the specified message type.

Restrictions

Cannot specify multiple hosts, imps, and so forth.

Cannot follow the reception of messages without looking at those currently in the file.

See Also

imp(4), implogd(8c)

implogd(8c)

Name

implogd – IMP logger process

Syntax

/etc/implogd [–d]

Description

The `implogd` program logs messages from IMP, placing them in the file `/usr/adm/implog.`

Entries in the file are variable length and each log entry has a fixed length header of the form:

```
struct sockstamp {
       short sin_family;
       u_short      sin_port;
       struct in_addr sin_addr;
       time_t sin_time;
       int    sin_len;
};
```

This is followed, possibly, by the message received from the IMP. Each time the logging process is started up, it places a time stamp entry in the file (a header with the `sin_len` field set to 0).

Restrictions

The logging process will catch only those messages from the IMP which are not processed by a protocol module. Thus the log should contain only status information such as ''IMP going down'' messages, ''host down'' and other error messages, and, perhaps, stray NCP messages.

See Also

imp(4p), implog(8c)

Name

inetd – internet service daemon

Syntax

/etc/inetd [**–d**] [*configfile*]

Description

The `inetd` daemon is the listener daemon for most of the internet service functions.

When `inetd` is started, it reads the configuration file specified (*configfile*) and opens a socket for each specified service.

When `inetd` receives a connection on a stream socket or a packet on a datagram socket, then `inetd` invokes the server specified in the configuration file to service the request. The server is given a socket descriptor of 0 for the service requested. The *configfile* is the configuration file specifying the services requiring the `inetd` daemon's services. If a configuration file is not specified, then `inetd` uses the default file, `/etc/inetd.conf.` The format of this file is described in `inetd.conf`(5). The configuration file is reread whenever `inetd` receives a hangup signal.

Options

–d Open all sockets with the debug option. The socket will be passed to the server with debug enabled.

Restrictions

The `inetd` daemon can only handle a limited number of services at any one time. This number is related to the maximum number of file descriptors that a process can have. If many services are needed, you should run multiple copies of `inetd`, each with its own individual configuration file.

Files

`/etc/inetd.conf`

See Also

inetd.conf(5)

Name

init – process control initialization

Syntax

/bin/init [*options*]

Description

The ULTRIX system invokes the `init` command as the last step in the boot procedure. The system normally then runs the automatic reboot sequence, as described in `reboot`(8). If `reboot` succeeds, the `init` command begins multiuser operation. If `reboot` fails, `init` begins single-user operation by giving the superuser a shell on the console.

You can use the `boot` command so that parameters are passed from the boot program to `init` so that multiuser operation begins immediately. When the superuser terminates the single-user shell (by pressing CTRL/D), `init` runs the `/etc/rc` command file without the reboot parameter. This command file performs housekeeping operations such as removing temporary files, mounting file systems, and starting daemons. For further information, see `reboot`(8).

In multiuser operation, `init` creates a process for each terminal port where a user may log in. To begin such operations, it reads the file `/etc/ttys`. For further information, see `ttys`(5). For each terminal that is marked ''on'' in the `ttys` file, `init` forks and invokes the command specified for the current line. The command is passed the name of the terminal as the last argument. The other arguments (if any) are specified after the command in the `ttys` file. Usually, the command is `getty`(8), but it may be any command.

The `getty` command reads the user's name and invokes `login` to log in the user and execute the shell.

Ultimately, the shell terminates because of an end-of-file. The end-of-file may be typed explicitly or generated as a result of hanging up on a terminal line. The main path of `init`, which has been waiting for such an event, wakes up and removes the appropriate entry from the file `utmp`, which records current users. The `init` command then makes an entry in `/usr/adm/wtmp`, which maintains a history of logins and logouts. The `wtmp` entry is made only if a user logged in successfully on the line. Then the appropriate terminal is reopened and `getty` is reinvoked.

The `init` command catches the `hangup` signal (signal SIGHUP) and interprets it to mean that the file `/etc/ttys` should be read again. The shell process on each line which used to be active in `ttys` but is no longer there is terminated; a new process is created for each added line; lines unchanged in the file are undisturbed. Thus it is possible to drop or add phone lines without rebooting the system by changing the `ttys` file and sending a `hangup` signal to the `init` process, using `kill -HUP 1`.

The `init` command terminates multiuser operations and resumes single-user mode if it receives a terminate (TERM) signal. That is, the superuser types `kill -TERM 1`. If there are processes outstanding which are deadlocked (due to hardware or software failure), `init` does not wait for them all to die, but times out after 30 seconds and prints a warning message.

If `init` receives a terminal stop signal (the superuser types `kill -TSTP 1`), `init` stops creating new processes and lets the system slowly die away. A later hangup will resume full multiuser operations, or a terminate will initiate a single user shell. This feature is used by `reboot`(8) and `halt`(8).

If `init` dies, the system will reboot itself automatically. If, at bootstrap time, the `init` process cannot be located, the system will loop in user mode at location 0x13.

Options

–a Specifies that the system should autoreboot to multiuser mode. This option is similar to specifying `auto` to the console prompt or specifying either `shutdown` with the **–r** option or `reboot` from the command line.

–s Specifies that the system should boot to single-user mode.

Diagnostics

WARNING: Something is hung (wont die); ps axl advised
The system is shutting down and `init` cannot kill a certain process. This usually occurs when a process cannot exit a device driver due to a persistent device error condition.

init: *'command tty'* **failing, sleeping.**
The `init` command tried to spawn a new process (use the `execve`(2) system call) for the *command* five times. Each time, the *command* failed. This may indicate that the *command* was invoked with invalid arguments. Check the `/etc/ttys` file for errors. This error message is printed at `syslog`(3) level LOG_ERR.

init: exec failed: cmd= *command reason*
The `init` commmand tried to spawn a new process using the `execve`(2) system call for the *command.* The `execve` failed. The *reason* is the explanation produced by the `perror`(3) routine for why the `execve` failed. This error message is printed at `syslog`(3) level LOG_ERR.

Files

`/etc/utmp`	Lists current system users
`/usr/adm/wtmp`	History of logins and logouts
`/etc/ttys`	The `init` command reads this file for a command to execute for the terminal line
`/etc/rc`	Command file executed by `init`

See Also

ttys(5), getty(8), rc(8), reboot(8)

Name

init – process control initialization

Syntax

/bin/init

Description

The ULTRIX system invokes the `init` command as the last step in the boot procedure. The system normally then runs the automatic reboot sequence, as described in `reboot`(8). If `reboot` succeeds, `init` begins multiuser operation. If `reboot` fails, `init` begins single-user operation by giving the superuser a shell on the console.

You can use the BOOT command so that parameters are passed from the boot program to `init` so that multi-user operation begins immediately. When the superuser terminates the single-user shell (by pressing CTRL/D), `init` runs the `/etc/rc` command file without the reboot parameter. This command file performs housekeeping operations such as removing temporary files, mounting file systems, and starting daemons. For further information, see `reboot(8)`.

In multi-user operation, `init` creates a process for each terminal port where a user may log in. To begin such operations, it reads the file `/etc/ttys`. For further information, see `ttys`(5). For each terminal that is marked ''on'' in the `ttys` file, `init` forks and invokes the command specified for the current line. The command is passed the name of the terminal as the last argument. The other arguments (if any) are specified after the command in the `ttys` file. Usually, the command is `getty`(8), but it may be any command.

The `getty` command reads the user's name and invokes `login` to log in the user and execute the shell.

Ultimately, the shell terminates because of an end-of-file. The end-of-file may be typed explicitly or generated as a result of hanging up on a terminal line. The main path of `init`, which has been waiting for such an event, wakes up and removes the appropriate entry from the file `utmp`, which records current users. The `init` command then makes an entry in `/usr/adm/wtmp`, which maintains a history of logins and logouts. The `wtmp` entry is made only if a user logged in successfully on the line. Then the appropriate terminal is reopened and `getty` is reinvoked.

`init` catches the `hangup` signal (signal SIGHUP) and interprets it to mean that the file `/etc/ttys` should be read again. The shell process on each line which used to be active in `ttys` but is no longer there is terminated; a new process is created for each added line; lines unchanged in the file are undisturbed. Thus, it is possible to drop or add phone lines without rebooting the system by changing the `ttys` file and sending a `hangup` signal to the `init` process, using `kill -HUP 1`.

The `init` command terminates multi-user operations and resumes single-user mode if it receives a terminate (TERM) signal. That is, the superuser types `kill -TERM 1`. If there are processes outstanding which are deadlocked (due to hardware or software failure), `init` does not wait for them all to die, but times out after 30 seconds and prints a warning message.

If `init` receives a terminal stop signal (the superuser types `kill -TSTP 1`), `init` stops creating new processes and lets the system slowly die away. A later hangup will resume full multi-user operations, or a terminate will initiate a single user shell. This feature is used by `reboot`(8) and `halt`(8).

If `init` dies, the system will reboot itself automatically. If, at bootstrap time, the `init` process cannot be located, the system will loop in user mode at location 0x13.

Diagnostics

WARNING: Something is hung (wont die); ps axl advised
The system is shutting down, and `init` cannot kill a certain process. This usually occurs when a process cannot exit a device driver due to a persistent device error condition.

init: '*command tty*' **failing, sleeping**
The `init` command tried to spawn a new process (use the `execve`(2) system call) for the *command* five times. Each time, the *command* failed. This may indicate that the *command* was invoked with invalid arguments. Check the `/etc/ttys` file for errors. This error message is printed at `syslog`(3) level LOG_ERR.

init: exec failed: cmd= *command reason*
The `init` command tried to spawn a new process using the `execve`(2) system call for the *command*. The `execve` failed. The *reason* is the explanation produced by the `perror`(3) routine for why the `execve` failed. This error message is printed at `syslog`(3) level LOG_ERR.

Files

`/etc/utmp`	List of current system users
`/usr/adm/wtmp`	History of logins and logouts
`/etc/ttys`	The `init` command reads this file for a command to execute for the terminal line
`/etc/rc`	Command file executed by `init`

See Also

ttys(5), getty(8), rc(8), reboot(8)

install-mh (8mh)

Name

install-mh – initialize the MH environment

Syntax

/usr/new/lib/mh/install–mh [–auto]

Description

When a user runs any MH program for the first time, the program will invoke `install-mh` (with the `-auto` switch) to query the user for the initial MH environment. The user does not invoke this program directly. The user is asked for the name of the directory that will be designated as the user's MH directory. If this directory does not exist, the user is asked if it should be created. Normally, this directory should be under the user's home directory, and has the default name of `Mail/`. After `install-mh` has written the initial `.mh_profile` for the user, control returns to the original MH program.

As with all MH commands, `install-mh` first consults the `$HOME` envariable to determine the user's home directory. If `$HOME` is not set, then the `/etc/passwd` file is consulted.

Files

`$HOME/.mh_profile`	User profile

Profile Components

Path: To set the user's MH directory

Context

With `-auto,` the current folder is changed to `inbox.`

Name

kdb_destroy – destroy the Kerberos master database

Syntax

/var/dss/kerberos/bin/kdb_destroy

Description

The `kdb_destroy` command removes the Kerberos master database by unlinking the `/var/dss/kerberos/dbase/principal.dir`, `/var/dss/kerberos/dbase/principal.pag`, and `/var/dss/kerberos/dbase/principal.ok` files.

Restrictions

This utility can be used only on the master database host.

Files

```
/var/dss/kerberos/dbase/principal.dir
/var/dss/kerberos/dbase/principal.pag
/var/dss/kerberos/dbase/principal.ok
```

See Also

kdb_init(8krb), kdb_edit(8krb), kstash(8krb), kdb_util(8krb)

kdb_edit(8krb)

Name

kdb_edit – Kerberos database editing utility

Syntax

/var/dss/kerberos/bin/kdb_edit [*-n*]

Description

You use the `kdb_edit` command to create or change principals stored in the Kerberos database. When you invoke `kdb_edit`, the command prompts you for the Kerberos database master key and verifies that the key is the same as the master key of the Kerberos database. If the *-n* option is used, the key is fetched from the master key file.

Once the master key is verified, `kdb_edit` begins a loop that prompts you for the principal and instance name to modify. If `kdb_edit` does not find an entry, you can create one. Once `kdb_edit` finds or creates an entry, you can set the password, expiration date, maximum ticket lifetime, and attributes of a principal.

The `kdb_edit` command displays, in brackets, the default values for the expiration dates, maximum ticket lifetimes, and attributes. You can select any default by pressing the return key. The `kdb_edit` command indicates that you have successfully created or changed an entry by displaying the message, "Edit O.K."

There is no default password for a principal. However, if you enter RANDOM as the password for a principal, `kdb_edit` selects a random Data Encryption Standard (DES) key for the principal.

Whenever the `ndbm` Kerberos database is changed by `kdb_edit`, the modification time of `/var/dss/kerberos/dbase/principal.ok` is updated. The `principal.ok` file indicates the age of the database.

Options

–n If you specify the command with the *-n* option, `kdb_edit` fetches the key from the master key file.

Files

`/var/dss/kerberos/dbase/principal.pag`

`/var/dss/kerberos/dbase/principal.dir`

`/var/dss/kerberos/dbase/principal.ok`

See Also

kdb_init(8krb), kstash(8krb), kdb_util(8krb), kdb_destroy(8krb)

Name

kdb_init – initialize the Kerberos master database

Syntax

/var/dss/kerberos/bin/kdb_init [*realm-name*] [*database-name*]

Arguments

realm-name	The realm of the Kerberos database.
database-name	A database specified so that the current `/var/dss/kerberos/dbase/principal.pag`, `/var/dss/kerberos/dbase/principal.dir`, and `/var/dss/kerberos/dbase/principal.ok` files are not overwritten.

Description

The `kdb_init` utility creates and initializes the Kerberos master database. The utility creates the database files: `/var/dss/kerberos/dbase/principal.dir`, `/var/dss/kerberos/dbase/principal.pag`, and `/var/dss/kerberos/dbase/principal.ok`. It also initializes the database by adding three database entries: the master database principal, a Kerberos default principal, the ticket-granting service principal (`krbtkt`), and the password changing principal, `changepw`.

The master database principal is the entry to the database itself. You cannot use or modify the database without the master database password. The Kerberos default principal provides a template for service principals.

The ticket-granting service, `krbtkt`, is used by Kerberos principals to obtain tickets to communicate with other Kerberos principals. The password-changing principal is not used.

If *realm-name* is omitted when you enter the command, `kdb_init` prompts for it. The program also prompts for the master database key. You cannot manipulate the database without this key.

By using *database-name*, you can create another database to prevent the current `principal.dir`, `principal.pag`, and `principal.ok` files from being overwritten.

After using `kdb_init` to set up the master database, you may want to use the `kstash(8krb)` utility to hide the master database password on the database host machine. This enables Kerberos administration programs to access and manipulate the master database, without needing the password to be entered manually.

Files

`/var/dss/kerberos/dbase/principal.pag`

`/var/dss/kerberos/dbase/principal.dir`

`/var/dss/kerberos/dbase/principal.ok`

See Also

kdb_util(8krb), kstash(8krb), kdb_edit(8krb), kdb_destroy(8krb)

Name

kdb_util – Kerberos database utility

Syntax

/var/dss/kerberos/bin/kdb_util *operation filename* [*database*]

Arguments

operation Function to perform on the database. The *operation* argument must be one of the following values:

load Converts the database in file *filename*, to `ndbm` format and overwrites the `ndbm`-formatted database specified.

dump
Converts the `ndbm`-formatted database into `krb_dbase(5krb)` form, and writes the result to file, *filename*.

slave_dump
Performs the same function as **dump**, and creates the file *filename*`.dump_ok` when finished. The *filename*`.dump_ok` is used by `kprop` to determine if a **slave_dump** is in progress, or if it did not complete successfully.

new_master_key
Prompts you for the old key of the Kerberos database as well as a new master key. It converts the `ndbm`-formatted database into `krb_dbase(5krb)` format and, at the same time, decrypts those sections of the database encrypted with the old master key and re-encrypts them with the new master key. The result is written to the file, *filename*.

filename The name of the source file for the **load** operation or the destination file for the operations: **dump**, **slave_dump**, and **new_master_key**.

database The name of the `ndbm`-formatted database. If the argument is not included, the Kerberos database is stored in files `/var/dss/kerberos/dbase/principal.dir`, `/var/dss/kerberos/dbase/principal.pag`, and `/var/dss/kerberos/dbase/principal.ok` by default.

Description

The `kdb_util` command allows the Kerberos administrator to perform several functions on the entire Kerberos database of a master or slave Kerberos server in one operation. The database argument specifies the name of the `ndbm`-formatted Kerberos database. The Kerberos database utility reads from and writes to the `ndbm`-formatted Kerberos database and, in addition, it reads from and writes to a file in `krb_dbase(5krb)` format: *filename*. A `krb_dbase(5krb)` file is an ASCII representation of a Kerberos database. The functions that can be specified by the *operation* argument are listed in the **Arguments** section.

Whenever the `ndbm` Kerberos database is changed by `kdb_util`, the modification time of `/var/dss/kerberos/dbase/principal.ok` is updated. The `principal.ok` file indicates the age of the database.

Files

`/var/dss/kerberos/dbase/principal.pag`

`/var/dss/kerberos/dbase/principal.dir`

`/var/dss/kerberos/dbase/principal.ok`

See Also

krb_dbase(5krb), kdb_init(8krb), kdb_edit(8krb), kdb_destroy(8krb), kstash(8krb)

Name

kdestroy – destroy Kerberos tickets

Syntax

/usr/bin/kdestroy [**–f**] [**–q**]

Description

The `kdestroy` utility destroys the user's active Kerberos authorization tickets by writing zeros to the file that contains them. If the ticket file does not exist, `kdestroy` displays a message to that effect.

After overwriting the file, `kdestroy` removes the file from the system. The utility displays a message indicating the success or failure of the operation. If `kdestroy` is unable to destroy the ticket file, the utility will issue a warning by making the terminal beep. The ticket file has the name, `/var/dss/kerberos/tkt/tkt`[*uid*] where *uid* is the user ID of the `kdestroy` process.

If your site does not provide a ticket-destroying mechanism, you can place the `kdestroy` command in your `.logout` file so that your tickets are destroyed automatically at logout.

Options

–f	Causes `kdestroy` to run without displaying the status message.
–q	Disables terminal beeping if `kdestroy` fails to destroy the tickets.

Restrictions

The `kdestroy` utility is useful only in environments with user-level authentication. ULTRIX Kerberos does not support user-level authentication.

Only the tickets in the user's current ticket file are destroyed.

Files

`/var/dss/kerberos/tkt/tkt`[*uid*]

See Also

kinit(8krb), klist(8krb)

Name

kerberos – the kerberos daemon

Syntax

/usr/etc/kerberos [**-p** *pause_seconds*] [**-a** *max_age*]
[**-l** *log_file*] [**-r** *realm*] [**-s**] [**-n**] [**-m**]

Description

The `kerberos` daemon is used by a Kerberos principal, X, to assist it in authenticating its identity to another Kerberos principal Y. In the ULTRIX environment, X would typically be an application running on one machine while Y would be an application running on another machine. Because X and Y run on separate machines, the authentication of X by Y and Y by X is not an easy task. If they ran on a single machine, A, the authentication of X could be performed easily by Y. All Y need do is ask A for the user ID of X. Since Y trusts the local machine, if the user ID of X is the user ID Y expects, then X must be X.

If Y were to authenticate X when X runs on a different machine, B, using the same user ID method, then Y would be forced to trust the machine B to provide a correct answer. The security of this method breaks down as soon as any one machine that Y is willing to trust is subverted by a hostile user. In addition, it breaks as soon as any machines that cannot be trusted by Y are allowed on the physical network to which A and B are connected. Hostile users that have control over these rogue machines can force them to produce messages that look as though they come from machine B.

The `kerberos` daemon serves as a single point of trust in a local area network (LAN). The authentication of X to Y depends upon the trust that both X and Y have in the `kerberos` daemon. X trusts the `kerberos` daemon to give Y only enough information to authenticate itself as Y to X, and Y trusts `kerberos` to give X only enough information to authenticate itself as X to Y. Y no longer needs to trust B to authenticate X.

If X were to authenticate itself to Y, X would first communicate with the `kerberos` daemon in order to obtain a ticket that would allow it to authenticate to Y. The ticket can be defined as the data that X needs to authenticate itself to Y. X passes the ticket to Y, along with other information, to authenticate itself to Y. Y then has the ability to send a message back to X in order to authenticate its identity to X.

There is one `kerberos` master daemon per LAN. The difference between a Kerberos master daemon and a Kerberos slave daemon is apparent in the way in which the Kerberos database on the machines on which they run is updated. The Kerberos database stores information about Kerberos principals. It stores, for instance, the Data Encryption Standard (DES) encryption key that is associated with each principal.

There is only one Kerberos database per LAN, to which updates to individual principal entries should be performed. This is the Kerberos master database. The `kerberos` daemon that runs on the machine which stores the Kerberos master database is the `kerberos` master daemon. All the other Kerberos databases in the LAN are periodically updated by `kprop`(8krb) and `kpropd`(8krb), based upon the data stored in the Kerberos master database. The machines that store this type of database run `kerberos` slave daemons.

A **realm** is the common name given to a group of principals. All principals stored in one Kerberos database belong to a single realm, and an individual `kerberos` daemon uses only one Kerberos database. So, a `kerberos` daemon only allows one principal in the realm to authenticate another principal in the realm. Inter-realm authentication is not supported in the ULTRIX version of Kerberos.

Options

–p Allows the user to select the number of seconds that the `kerberos` daemon will pause, *pause_seconds*, after it has encountered an unrecoverable error, and before it exits. This time interval must be between five minutes (300), and one hour (3600). If neither this option nor the **-s** option is used, the `kerberos` daemon will pause forever before exiting.

–a Allows the user to specify the age in seconds, *max_age*, above which the Kerberos database should be considered too old for a Kerberos slave server to use. The `kerberos` daemon determines the age of the Kerberos database by comparing the last modification time of the `/var/dss/kerberos/dbase/principal.ok` file with the current time. The `principal.ok` file is modified every time the database is changed. Since a Kerberos slave server receives its database in whole from the Kerberos master, this option specifies the maximum amount of time allowed between database transfers. The time value must be between one hour (3600) and three days (259200). If neither this option nor the **-s** option is used, the maximum age of the database is infinite.

–l Allows the user to select a different file, *log_file*, into which the `kerberos` daemon will place Kerberos log messages. If neither this option nor the **-s** option is used, the *log_file* value is set to `/var/dss/kerberos/log/kerberos.log`.

–r Allows the user to change the name of the realm, *realm*, for which the `kerberos` daemon will serve information. If no realm name is specified with the **-r** option, the `kerberos` daemon will server the realm of which the local host is a member.

–s Allows the user to tell the `kerberos` daemon to use the default values for *pause_seconds*, *max_age*, and *log_file* of a slave server. If *max_age* has not been set with the **-a** option, the *max_age* value is set to the slave server default of one day (86400). If the *pause_seconds* value has not been set with the **-p** option, the *pause_seconds* value is set to the slave server default of 5 minutes (300). If the *log_file* value has not been set with the **-l** option, the *log_file* value is set to the slave server default, `/var/dss/kerberos/log/kerberos_slave.log`. Use of the **-s** option is equivalent to using the following list of options with the `kerberos` daemon:

```
-a 86400 -p 300 -l /var/dss/kerberos/log/kerberos_slave.log
```

–n Allows the user to tell the `kerberos` daemon that the maximum age of the Kerberos database should be infinite. This option is only useful if the **-s** option has been selected by the user, but the maximum age of the database should not be equal to the slave default (300), but should be infinite. This option also overrides the **-a** option.

–m Allows the user to run the `kerberos` daemon in manual mode. This implies that the master key of the Kerberos database will be input from `stdin`. If this option is not used, the master key of the Kerberos database is read from the data file `kstash(8krb)` placed in the system.

See Also

kdb_init(8krb), kdb_util(8krb), kdb_edit(8krb), kdb_destroy(8krb), kerberos(3krb), kprop(8krb) kpropd(8krb)

Name

kgconv – convert a dump from kgmon (kgdump.out) to gprof format (gmon.out)

Syntax

kgconv [**–d**] [**–n***cpunumber*] [*dumpfile*] [system]

Description

The `kgconv` command is used to convert the dump file (kgdump.out) produced by `kgmon` to the format required by `gprof`. The output file is gmon.out, which can be directly used by `gprof`. If the **–n** flag is not used, kgconv dumps the cumulative profile data about all the processors in the system.

Options

–n*cpunumber*	Converts profile data for the specified processor only.
–d	Uses the *dumpfile* specified as the next argument, instead of kgdump.out as the input file.

Files

`/vmunix`	default system
`./kgdump.out`	default input dump file
`./gmon.out`	output file used by `gprof`

See Also

kgmon(8), gprof(1), config(8)

kgmon(8)

Name

kgmon – generate a dump of the operating system's profile buffers

Syntax

/etc/kgmon [*options*] [*system*] [*memory*]

Description

The `kgmon` command is used when profiling the operating system. When no arguments are supplied, `kgmon` indicates the state of operating system profiling as running, off, or not configured. For further information, see `config`(8). If the `-p` option is specified, `kgmon` extracts profile data from the operating system and produces a `kgdump.out` file suitable for later analysis by `gprof`.

The `kgdump.out` file is first converted to a format suitable for `gprof` by using the `kgconv` filter. The `kgmon` command dumps the kernel profiling data for all the processors in the system. The `kgconv` command is used to create a `kgmon.out` file suitable for analysis by `gprof` for any or all the processors in the system.

Options

–b	Resumes the collection of profile data.
–h	Stops the collection of profile data.
–p	Dumps the contents of the profile buffers into a `kgdump.out` file.
–r	Resets all the profile buffers. If the `-p` option is also specified, the `kgdump.out` file is generated before the buffers are reset.

If neither `-b` nor `-h` is specified, the state of profiling collection remains unchanged. For example, if the `-p` option is specified and profile data is being collected, profiling will be momentarily suspended, the operating system profile buffers will be dumped, and profiling will be immediately resumed.

Diagnostics

Users with only read permission on `/dev/kmem` cannot change the state of profiling collection. They can get a `kgdump.out` file with the warning that the data may be inconsistent if profiling is in progress.

Files

`/vmunix`	Default system
`/dev/kmem`	Default memory

See Also

gprof(1), config(8), kgconv(8)

Name

kinit – Kerberos login utility

Syntax

/usr/bin/kinit [**–irv**]

Description

You use the `kinit` command to log into the Kerberos authentication and authorization system. You also use the `kinit` command when your original tickets have expired. When you use the `kinit` command without options, the utility prompts for a username and a Kerberos password and attempts to authenticate to the local Kerberos server.

If Kerberos authenticates you correctly, `kinit` retrieves your initial ticket and puts it in the ticket file specified by the KRBTKFILE environment variable. If you have not defined this variable, the ticket is stored in the file `/var/dss/kerberos/tkt/tkt[`*uid*`]`.

Make sure you use the `kdestroy(8krb)` command to destroy any active tickets before ending your login session. You may want to put the `kdestroy` command in a `.logout` file so that all tickets are destroyed automatically when you log out. Only registered Kerberos users can use the Kerberos system.

Options

–i	Causes `kinit` to prompt you for a Kerberos instance.
–r	Causes `kinit` to prompt you for a Kerberos realm. This option lets you authenticate yourself with a remote Kerberos server.
–v	Initiates verbose mode. This causes `kinit` to print the name of the ticket file used and a status message indicating the success or failure of your login attempt.

Restrictions

Although user-level authentication is not supported, `kinit` is useful for testing the installation of Kerberos functionality, by determining if a newly installed principal can obtain a ticket-granting ticket. For example, to determine if the `named` running on machine X can obtain its ticket-granting ticket, you can run `kinit`, input the principal name, `named`, the instance, X, and the password of `named`. If `kinit` succeeds, then Kerberos is correctly installed on machine X.

The `-r` option has not been fully implemented.

Files

`/var/dss/kerberos/tkt/tkt[`*uid*`]`

See Also

kdestroy(8krb), klist(8krb)

Name

klist – lists currently held Kerberos tickets

Syntax

/usr/bin/klist [**–s** | **-t**] [**–file** [*filename*]] [**–srvtab**]

Arguments

filename	The name of the Kerberos ticket file.

Description

The `klist` command allows you to print the name of the ticket file, the identity of the principal requesting the tickets (as listed in the ticket file), and the principal names of all the Kerberos tickets currently held by the user (along with the issue and expiration times for each authenticator). Principal names are listed in the form:

```
name.instance@realm
```

The period (.) is omitted if the instance is null, and the at sign (@) is omitted if the realm is null.

The `klist` command also enables you to print the entries in the `srvtab` file. If the **-srvtab** option is selected, `klist` will print the service name, instance name, realm name, and key version of all services listed in the `srvtab` file.

Options

–s Suppresses the printing of the issue and expiration times, the name of the ticket file, or the identity of the principal.

–t Checks for the existence of an unexpired ticket-granting-ticket in the ticket file. If one is present, `klist` exits with status of zero (0). Otherwise, it exits with status 1. No output is generated when this option is specified.

–file Causes the following argument to be used as the ticket file. Otherwise, the file `/var/dss/kerberos/tkt/tkt[`*uid*`]` is used, where `uid` is the user ID of the `klist` process.

–srvtab Indicates that `srvtab` data should be printed. If the **-file** switch is not used, the `srvtab` data is read from the default `srvtab` file, `/etc/srvtab`.

Restrictions

User-level authentication is not supported. However, by naming the file `tkt.login` with the `-file` option, you can look at the tickets generated by `login`.

Files

`/etc/srvtab`	Default `srvtab` file
`/etc/krb.conf`	To get the name of the local realm
`/var/dss/kerberos/tkt/tkt`[*uid*]	The default ticket file
`/var/dss/kerberos/tkt/tkt.login`	The file containing tickets generated by `.login`

See Also

kinit(8krb), kdestroy(8krb)

Name

kprop – Kerberos utility

Syntax

/var/dss/kerberos/bin/kprop *database slaves_file* [**–force**] [**–safe** | **clear**] [**–realm** *realm_name*]

Description

The `kprop` daemon runs on a Kerberos master and propagates the Kerberos database to the Kerberos slaves, where it is received by the waiting `kpropd` daemon.

The first parameter, *database*, is the name of the file out of which data is extracted. This file is not the `ndbm`-formatted Kerberos database, `/var/dss/kerberos/dbase/principal`. See the `ndbm(3)` reference page for more information. The *database* is a file created by the `kdb_util slave_dump` command. It is an ASCII representation of the Kerberos database (see the reference page for `krb_dbase(5krb)`.

The second parameter that must be supplied is *slaves_file*, the name of the file on the Kerberos master that lists the Kerberos slaves to which `kprop` propagates the Kerberos master database. The `slaves_file` is created in `krb_slaves(5krb)` format.

The Kerberos utility first determines whether the ASCII Kerberos database, *database*, was correctly dumped by `kdb_util`. It accomplishes this by determining if *database* is older than the `database.dump_ok` file created by `kdb_util` during the `slave_dump` operation. If it is older, the dump did not succeed or is not yet finished. If the dump did not complete successfully or has not yet completed, the master database is not transferred to any Kerberos slave. Otherwise, `kprop` determines, for each slave server listed in the `slaves_file`, whether or not the database has changed since the last successful transfer to the slave. It determines this for slave server `cactus` by comparing the modification time of the `/etc/cactus-last-prop` file with the modification time of *database*. If the `/etc/cactus-last-prop` file is newer, then the database, *database*, need not be transferred to `cactus`. Finally, `kprop` propagates the database to those servers which need a new copy of the database and updates the modification time of the `/etc/server-last-prop` file for these slave servers.

Options

–safe Specifies that the data sent over the network is guaranteed to be authenticated at the destination and protected against modifications in transit. That is, `kprop` and `kpropd`, which are Kerberos principals, become Kerberos-authenticated to each other and send messages formatted by `krb_mk_safe`. For more information about `krb_mk_safe`, refer to the on-line reference page, `kerberos(3krb)`.

–clear Specifies that all data should be sent in cleartext (unencrypted). This switch is useful when first setting up the Kerberos environment.

–realm Specifies the realm name that you are in. If this option is not used, the *realm_name* is given in the `/etc/krb.conf` file. (See the `krb.conf(5krb)` reference page for more information.)

–force Forces the `kprop` on the Kerberos master to propagate the Kerberos database to the Kerberos slaves, even if there are no recent changes to the database. Without the force flag, the Kerberos database is not propagated if the database file has not changed since the last successful transfer.

Restrictions

The Kerberos utility does not support the transfer of encrypted data.

Files

```
/usr/var/dss/kerberos/dbase/principal.dir
/usr/var/dss/kerberos/dbase/principal.pag
/usr/var/dss/kerberos/dbase/principal.ok
/etc/krb.conf
```

See Also

kpropd(8krb), krb.conf(5krb), kdb_util(8krb), krb_slaves(5krb), krb_dbase(8krb)

Name

kpropd – Kerberos utility

Syntax

/usr/etc/kpropd *output_file* [**–d** *krb_database*] [**–l** *log_file*]
[**–r** *realm_name*] [**–s** *srvtab_file*]

Description

The `kpropd` daemon runs on a Kerberos slave and waits to receive the Kerberos database propagated from a `kprop` process on a Kerberos master. The first parameter, *output_file*, that you must supply to the `kpropd` daemon is the name of a database file in which data will be placed when it comes over the network.

The `kpropd` utility executes the `kdb_util(8krb)` utility, which loads the database from the file specified in *output_file,* puts it in `ndbm` format, and copies it into the Kerberos database in the `/var/dss/kerberos/dbase` directory.

Options

–r Specifies the receiver realm for which data is accepted; `/etc/krb.conf` specifies the default. (See the `krb.conf(5krb)` reference page for more information.)

–s Specifies the service table (`srvtab`) file from which to read the password of the `kpropd` daemon, because a password cannot be entered manually when `kpropd` is running as a daemon. The default is `/etc/srvtab`.

–d Specifies the primary Kerberos database file of a Kerberos slave. This file receives a new or updated database propagated from the Kerberos master. The default is the `ndbm` database in the directory, `/usr/var/dss/kerberos/dbase`. The `ndbm` files are: `principal.dir`, `principal.ok`, and `principal.pag`.

–l This option specifies the name of the log file to use. The default log file is `/var/dss/kerberos/log/kpropd.log`.

Restrictions

The `kpropd` command does not support the transfer of encrypted data.

If the `/var/dss/kerberos/bin` directory is not included in the PATH environment variable of the process that runs `kprop`, then `kprop` will fail because it cannot locate `kdb_util`.

Files

`/usr/var/dss/kerberos/dbase/principal.ok`

`/usr/var/dss/kerberos/dbase/principal.pag`

`/usr/var/dss/kerberos/dbase/principal.dir`

See Also

kprop(8krb), kdb_util(8krb), krb.conf(5krb)

Name

kstash – hide the Kerberos master database key for automatic retrieval

Syntax

/var/dss/kerberos/bin/kstash

Description

The `kstash` administration utility stores the Kerberos master database password in a hidden place on the host machine of the database. The password is then available to other administration programs so that they can access and manipulate the database, without needing the password to be entered manually.

In general, `kstash` is used during the initial setup of a Kerberos system. The installer initializes the master database with `kdb_init`(8krb) and then uses `kstash` to hide the master database key. If someone changes the master key, you must invoke `kstash` again to store the new key.

Restrictions

This utility can only be used on Kerberos master and slave hosts.

See Also

kdb_init(8krb), kdb_edit(8krb), kdb_destroy(8krb), kdb_util(8krb)

Name

la75of – LA75 dot matrix printer filter

Syntax

/usr/lib/lpdfilters/la75of [**–c**] [**–n***login*] [**–h***host*] [**–w***width*] [**–l***num*] [**–i***indent*] [**accounting file**]

Description

The `la75of` filters text data destined for LA75 compact dot matrix printers. It handles the device dependencies of the printer and performs accounting functions. As each job completes, the filter writes the accounting records to the file specified by the **af** field in `/etc/printcap`. The `la75of` can handle plain ASCII files as well as files that have been preprocessed by nroff. The filter correctly handles nroff control sequences for underlining, superscripting and subscripting.

You can specify the `la75of` filter in both the **of** and the **if** fields of the `/etc/printcap` file. For further information, see `printcap`(5). When you specify both fields, the **of** filter prints the banner page only; when **of** stops, the **if** filter gains access to the printer and maintains accounting information.

If you specify the **of** field only, the filter prints the banner page then stops and restarts to maintain accounting information.

If you specify the **if** field only, the banner page is sent directly to the printer. This is not a problem for most impact printers.

For a more detailed discussion on filters see the article, ''Line Printer Spooler Manual'' in the *ULTRIX Supplementary Documents, Volume 2: Programmer.*

Options

The arguments passed to the filter depend on its use. The **of** filter is called with the following arguments:

la75of –w*width* **–l***length*

The *width* and *length* values come from the **pw** and **pl** fields in the `/etc/printcap` database. The **if** (or restarted **of**) filter is passed the following arguments:

la75of –c –n*login* **–h***host* **–w***width* **–l***num* **–i***indent* **accounting file**

The **–c** flag is optional and is supplied only when control characters are to be printed. That is, when the **–l** option of `lpr`(1) is used to print the file. The **–w** and **–l** arguments are the same as for the **of** filter but might have different values if the **–w** and/or **–z** options of `lpr`(1) were used to print the file. The **–n** and **–h** arguments specify the login name and host name of the job owner. These arguments are used to record accounting information. The **–i** option specifies the amount of indentation to be used. The last argument is the name of the accounting file specified from the **af** field in the `/etc/printcap` database.

Diagnostics

The **lf** field (default /dev/null) in the `/etc/printcap` database specifies the error logging file name.

Files

`/etc/printcap`
Printer capabilities database

`/dev/lp?`

See Also

lpr(1), pr(1), printcap(5), lpd(8), MAKEDEV(8), pac(8)
"Line Printer Spooler Manual," *ULTRIX Supplementary Documents, Volume 2: Programmer*

Name

lcg01of – LCG01 Color Printing System Filter

Syntax

/usr/lib/lpdfilters/lcg01of [**–c**] [**–n***login*] [**–h***host*] [**–w***width*] [**–l***num*] [**–i***indent*] [**accounting file**]

Description

The `lcg01of` filter is used to filter text data destined for the LCG01 Color Printing System. The filter handles the device dependencies of the printers and performs accounting functions. Accounting records are written to the file specified by the **af** field in `/etc/printcap` at the time of completion for each job.

The filter can handle plain ASCII files, nroff files, and sixel files. The printer also supports the following graphic protocols: ReGIS, NAPLPS, and GIDIS. To print using these protocols, the xf(8) filter should be invoked using the **lpr -x** option. The `lcg01of` filter translates nroff control sequences for underlining, superscripting, and subscripting into the correct LCG01 control sequences.

The `lcg01of` filter can be the specified filter in both the **of** and the **if** fields in the `/etc/printcap` file. For further information, see `printcap`(5). When both fields are specified, the **of** filter is used only to print the banner page. It is then stopped to allow the **if** filter access to the printer. The **if** filter maintains accounting information.

If the **of** field is the only one specified, the filter is used to print the banner page. It is then stopped and restarted. This allows the **of** filter to be used to maintain accounting information.

If the **if** field is the only one specified, the banner page is sent directly to the printer. If the printer has been left in an undetermined state, the banner page may not print correctly.

The best results are obtained when the filter is specified in both the **of** and **if** fields. For a more detailed discussion on filters, see the ''Line Printer Spooler Manual'' in the *ULTRIX Supplementary Documents, Volume 2: Programmer.*

Options

The `lcg01of` filter ignores the arguments passed to it by the line printer daemon, lpd(8). The **of** filter is called with the following arguments:

lcg01of –w*width* **–l***length*

The *width* and *length* values come from the **pw** and **pl** fields in the `/etc/printcap` database.

The **if** (or restarted **of**) filter is passed the following arguments:

lcg01of –c –n*login* **–h***host* **–w***width* **–l***num* **–i***indent* **accounting file**

Diagnostics

The **lf** field (default /dev/null) in the `/etc/printcap` database specifies error logging file name.

Files

`/etc/printcap`
Printer capabilities database

`/dev/lp?`

See Also

lpr(1), pr(1), printcap(5), lpd(8), MAKEDEV(8), pac(8)
''Line Printer Spooler Manual,'' *ULTRIX Supplementary Documents, Volume 2: Programmer*

Name

lcp – Local Area Transport server (LAT) control program

Syntax

/etc/lcp [*options*]

Description

The LAT control program, `lcp`, provides the essential functions to control and manage LAT terminal service. It allows you to start and stop LAT service, modify and display configuration characteristics, and display and set error counters to zero. The following command is usually included in the `/etc/rc.local` file to automatically restart LAT service during reboot:

```
lcp -s
```

Options

The following options can be set with `lcp`:

–s Starts LAT service. Enables connections from LAT terminal servers to host. If LAT parameters have not been set, they take on default values specified in the `-r` option.

–r Resets LAT parameters to the following default values:
multicast timer: 30 seconds
nodename: hostname
node description: ''ULTRIX''
servicename: hostname
service description: ''ULTRIX LAT SERVICE''

–g Sets groups to *grp1*, *grp2*, ... , *grpn*. A number or string of numbers can be used to set groups. The numbers used for each group must be less than or equal to 255. For example:

```
lcp -g 128

or

lcp -g 56,5,102,10,20,30,40,50,60,...,
110,150,200,210,255
```

A string of groups must be separated by commas with no spaces. If the string exceeds the width of the screen it must wrap over to the following line. You cannot use a backslash (\) or a carriage return to break a string.

–h Sets a list of ttys (next argument) as being available only for host-initiated connections. A string of ttys must be separated by commas with no spaces. If the string exceeds the width of the screen, it must wrap over to the following line. You cannot use a backslash (\) or a carriage return to break a string. Each tty can optionally be associated with a specific port on a specific terminal server by following the tty name by the name of the server and port, separated by colons. For example, the following command associates tty15 with the port named "PORT7" on the terminal server named "LAT_SERVER".

```
/etc/lcp -h /dev/tty15:LAT_SERVER:PORT7
```

–H Sets a list of ttys (next argument) as being available only for terminal server initiated connections. A string of ttys must be separated by commas with no spaces. If the string exceeds the width of the screen it must wrap over to the following line. You cannot use a backslash (\) or a carriage return to break a string.

–m Sets multicast transmission timer to the specified time (next argument). A node advertises its presence to the LAT servers by sending out a multicast message over the network. The *time* variable sets the interval between transmissions. Valid intervals range from 10 to 255 seconds, with a default of 30 seconds.

–n Sets node to the specified name (next argument). Specifies the name that your node will be known by. Although the Terminal Server keeps track of nodes without an associated nodename, a LAT node must have a nodename in order for a terminal user to establish a connection. A node can have a list of associated services and service ratings, specified by up to 16 alphanumeric characters. Dollar ($) and underscore (_) characters are valid, but leading underscores and trailing colons are removed. The nodename must be unique on the Ethernet. The Terminal Server displays the nodename as a service if you type the `show services` command, and as a node if you type the `show nodes` command. The nodename default is the hostname.

–N Sets node description to the specified message (next argument). The node description allows for a short message to be displayed to LAT users, providing news or additional node information. Specify up to 64 alphanumeric characters. Dollar ($) and underscore (_) characters are valid, but leading underscores and trailing colons are removed. Leading ampersands (&) are not allowed. The default is ''ULTRIX''. You must enclose the string in double quotation marks (" ") if it contains one or more spaces.

–v A service node advertises one or more services. By default the single service offered is the node name. For example, node `microv` by default offers service `microv`. The `-v` option lets you change this default. It also lets you cause the service node to advertise more than one service and associate a given set of LAT ttys with each service by appending a list of minor device numbers to the service name. For example, the following command causes the node to offer two additional services, "SERV1" and "SERV2":

```
/etc/lcp -v microv -v SERV1:/dev/tty15,/dev/tty16\
-v SERV2:/dev/tty17,/dev/tty18,/dev/tty19
```

Devices tty15 and tty16 are used for SERV1. Devices tty17, tty18, and tty19 are used for SERV2. All other LAT ttys are used for the default service, `microv`.

Note that normal LAT service is always associated with the first -v option to appear. Therefore, if you wish to advertise additional services you must define the service name for normal lat, even if you still want the default service name, as in the above example.

Every time you issue a new lcp command with the -v option, the new set of services that you define completely replaces any previously defined services. To discontinue a previously defined service, reissue lcp with the -v option without specifying that service. For example, to discontinue the SERV1 service, use the following command:

```
/etc/lcp -v microv \
-v SERV2:dev/tty17,dev/tty18,dev/tty19
```

–V Sets service description to specified message (next argument). It can be up to 64 characters in length. You must enclose the string in double quotation marks (" ") if it contains one or more spaces. If you are defining multiple services a given description applies to the service name defined by the corresponding -v option (the first -V option corresponds to the first -v option, and so on). For example, the following command associates the description "ULTRIX LAT service" with the service name microv and description "service 1" with service name "SERV1".

```
/etc/lcp -v microv -v SERV1:dev/tty15,dev/tty16\
-V "ULTRIX LAT service" -V "service 1"
```

–t Stops LAT service. Disable connections from LAT terminal servers to host.

–d Displays LAT characteristics. Shows the LAT parameters at their current setting. The following is an example of the output of the -d option:

```
% /etc/lcp -d

Node name: NODE      Service name: NODE
Node Identification: ULTRIX LAT service
Service Identification: ULTRIX
Groups: 0
Multicast timer: 30 seconds
LAT version: 5 eco: 0  LAT Protocol is active
```

–z Reinitializes (zeroes out) error counters. To test system performance over a period of time, zero the counters and observe the information that accumulates.

–c Displays error counters in vertical format. If an interval also is specified (next argument), displays error counters in horizontal format every interval seconds. The following is an example of the output of the -c option:

```
% /etc/lcp -c
67413 Frames received (rcv)
   32 Duplicate frames received (rcvdup)
89005 Frames transmitted (xmit)
   62 Retransmissions (rexmit)
    0 Illegal messages received (illmesg)
    0 Illegal slots received (illslots)
```

interval	Continuously displays error counters in horizontal format, waiting *interval* seconds between each iteration. Quit by sending a keyboard interrupt. The following is an example of the output of the interval option:

```
% /etc/lcp 10
rcv      rcvdup   xmit     rexmit   illmesg  illslots
67474    32       89066    62       0        0
67483    32       89067    62       0        0
67491    32       89073    62       0        0
67502    32       89089    62       0        0
```

–p	Shows which LAT server/port combination a specific LAT tty device is connected to. For example the following command displays which terminal server and port are associated with tty15.

```
/etc/lcp -p /dev/tty15
```

Restrictions

The service and node names cannot be more than 16 characters long.

The user must have read and write access to a terminal.

Error Counters

The meaning of each error counter is explained below.

rcv	Number of Ethernet LAT messages
rcvdup	Number of duplicate messages received (normally indicates a system slowdown)
xmit	Number of transmitted Ethernet LAT messages
rexmit	Number of transmit frames that are sent more than once
illmesg	Number of bad messages flagged by the LAT driver
illslots	Number of bad transmission slots flagged by the driver

Diagnostics

Messages indicating that user is not privileged or that LAT service is not loaded

Files

```
/etc/ttys
/etc/rc.local
```

See Also

lta(4), ttys(5)

Name

lg02of – LG02 Matrix Line Printer

Syntax

/usr/lib/lpdfilters/lg02of [**–c**] [**–n***login*] [**–h***host*] [**–w***width*] [**–l***num*] [**–i***indent*] [**accounting file**]

Description

The `lg02of` filter is used to filter text data destined for the LG02 line printer. The filter handles the device dependencies of the printers and performs accounting functions. Accounting records are written to the file specified by the **af** field in `/etc/printcap` at the time of completion for each job.

The filter can handle plain ASCII files and files that have been preprocessed by nroff. The `lg02of` filter translates nroff control sequences for underlining, superscripting and subscripting into the correct LG02 control sequences.

The `lg02of` filter can be the specified filter in both the **of** and the **if** fields in the `/etc/printcap` file. For further information, see `printcap`(5). When both fields are specified the **of** filter is used only to print the banner page. It is then stopped to allow the **if** filter access to the printer. The **if** filter maintains accounting information.

If the **of** field is the only one specified the filter is used to print the banner page. It is then stopped and restarted. This allows the **of** filter to be used to maintain accounting information.

If the **if** field is the only one specified the banner page will be sent directly to the printer. If the printer has been left in an undetermined state the banner page may not print correctly.

The best results will be obtained when the filter is specified in both the **of** and **if** fields. For a more detailed discussion on filters see the ''Line Printer Spooler Manual'' in the *ULTRIX Supplementary Documents, Volume 2: Programmer.*

Options

The arguments passed to the filter depend on its use. The **of** filter is called with the following arguments:

lg02of –w*width* **–l***length*

The *width and length* values come from the **pw** and **pl** fields in the `/etc/printcap` database. The **if** (or restarted of) filter is passed the following arguments:

lg02of –c –n*login* **–h***host* **–w***width* **–l***num* **–i***indent* **accounting file**

The **–c** flag is optional, and supplied only when control characters are to be printed, that is, when the **–l** option of `lpr`(1) is used to print the file. The **–w** and **–l** arguments are the same as for the **of** filter, however, they may have different values if the **–w** and/or **–z** options of `lpr`(1) were used to print the file. If the value of the width is greater than 80 the job will be printed in landscape mode with a slightly smaller font. The length of the page is assumed to be 66 regardless of the length specified. The **–n** and **–h** arguments specify the login name and host name of the job owner.

These arguments are used to record accounting information. The **–i** option specifies the amount of indentation to be used. The last argument is the name of the accounting file specified from the **af** field in the `/etc/printcap` database.

Diagnostics

Are sent to wherever `/etc/printcap` specifies with the **lf** field.

Files

```
/etc/printcap
/dev/lp?
```

See Also

lpr(1), pr(1), printcap(5), lpd(8), MAKEDEV(8), pac(8)
''Line Printer Spooler Manual,'' *ULTRIX Supplementary Documents, Volume 2: Programmer*

Name

lj250of – LJ250 Companion Color Printer Filter

Syntax

/usr/lib/lpdfilters/lj250of [**–c**] [**–n***login*] [**–h***host*] [**–w***width*] [**–l***num*] [**–i***indent*] [**accounting file**]

Description

The `lj250of` filter is used to filter text data destined for the LJ250 Companion Color printer. The filter handles the device dependencies of the printers and performs accounting functions. Accounting records are written to the file specified by the **af** field in `/etc/printcap` at the time of completion for each job.

The filter can handle plain ASCII files, nroff files, and sixel files. If the printer is in HP HCL mode, the `xf`(8) filter should be invoked using the **lpr -x** option. The `lj250of` filter translates nroff control sequences for underlining, superscripting, and subscripting into the correct LJ250 control sequences.

The `lj250of` filter can be the specified filter in both the **of** and the **if** fields in the `/etc/printcap` file. For further information, see `printcap`(5). When both fields are specified the **of** filter is used only to print the banner page. It is then stopped to allow the **if** filter access to the printer. The **if** filter maintains accounting information.

If the **of** field is the only one specified, the filter is used to print the banner page. It is then stopped and restarted. This allows the **of** filter to be used to maintain accounting information.

If the **if** field is the only one specified, the banner page is sent directly to the printer. If the printer has been left in an undetermined state the banner page may not print correctly.

The best results are obtained when the filter is specified in both the **of** and **if** fields. For a more detailed discussion on filters, see the ''Line Printer Spooler Manual'' in the *ULTRIX Supplementary Documents, Volume 2: Programmer.*

Options

The `lj250of` filter ignores the arguments passed to it by the line printer daemon, lpd(8). The **of** filter is called with the following arguments:

lj250of –w*width* **–l***length*

The *width* and *length* values come from the **pw** and **pl** fields in the `/etc/printcap` database.

The **if** (or restarted **of**) filter is passed the following arguments:

lj250of –c –n*login* **–h***host* **–w***width* **–l***num* **–i***indent* **accounting file**

Diagnostics

The **lf** field (default /dev/null) in the `/etc/printcap` database specifies error logging file name.

Files

`/etc/printcap`
Printer capabilities database

`/dev/lp?`

See Also

lpr(1), pr(1), printcap(5), lpd(8), MAKEDEV(8), pac(8)
''Line Printer Spooler Manual,'' *ULTRIX Supplementary Documents, Volume 2: Programmer*

llbd(8ncs)

Name

llbd – Local Location Broker Daemon

Syntax

/etc/ncs/llbd [**–version**]

Description

The Local Location Broker Daemon (`llbd`) is part of the Network Computing System (NCS). It manages the Local Location Broker (LLB) database, which stores information about RPC-based server programs running on the local host.

A host must run `llbd` if it is to support the Location Broker forwarding function or to allow remote access (for example, by the `lb_admin` tool) to the LLB database. In general, any host that runs an RPC-based server program must run an `llbd,` and `llbd` must be running before any such servers are started. Additionally, any network supporting RPC activity should have at least one host running a Global Location Broker Daemon (`nrglbd`).

On ULTRIX systems, `llbd` is typically started by a line in `/etc/rc` such as the following:

```
/etc/ncs/llbd& echo -n ' llbd' > /dev/console
```

NOTE

If your system contains more than one broadcast interface, the RPC software uses only the broadcast interface associated with the `/bin/hostname` value in your `/etc/rc.local` file.

Options

–version Display the version of the Network Computing Kernel (NCK) that this `llbd` belongs to, but do not start the daemon. (NCK is part of the Network Computing System (NCS) on which DECrpc is based.)

See Also

lb_admin(1ncs), nrglbd(8ncs)
Guide to the Location Broker

Name

lmf – License Management Facility (LMF)

Syntax

/etc/lmf [**–d** *dir*] [*command* [*argument...*]]

Description

You can use the `lmf` utility to maintain a file of registered software licenses. The file is called the License Database (LDB) and is derived from Product Authorization Key (PAK) information. You maintain the LDB by using the commands provided by the `lmf` utility (these are described in the LMF Commands section).

You can also use the `lmf` utility to keep the kernel cache updated. The kernel cache contains the active license information which is read by license checking functions in products that provide full LMF support. The license checking functions ensure that a product has a valid license before making the product available for use on the system.

For more information about the LMF, see the *Guide to Software Licensing.*

When you use the LMF commands you can type them on a single line, for example:

```
# lmf register
```

or you can enter the `lmf` utility and type the commands after the prompt, for example:

```
# lmf
lmf> register
```

You can abbreviate the commands, but you must ensure that the abbreviation is not ambiguous. For example, you could abbreviate `lmf register` to `lmf reg` but not `lmf re`.

If you need to use the *product*, *producer* or *authorization* arguments, they must be specified exactly as they are on the PAK. Use these arguments if the LDB contains more than one record for a given *product* name.

Options

–d *dir*
: Defines the directory containing the LDB file and the history file. If you do not use the **–d** option, the default LMF directory (`/usr/var/adm/lmf`) is used. The **–d** option allows you to have more than one LDB on your system.

LMF Commands

There are three types of `lmf` commands:

- Information commands, which you use to monitor your licensing actions, and the status of the LDB and kernel cache
- LDB maintenance commands, which you use to modify the contents of the License Database
- Service commands, which you use to communicate with the kernel cache and may have an immediate effect on users' access to licensed software

Information Commands

help [*command*]

Prints the syntax of the command specified in the argument list. If no command is specified, `lmf help` prints a list of all the recognized commands.

exit

Exits from the `lmf` utility. You can also press CTRL/D to leave the utility.

list [**full**] [*source*] [**for** *product* [*producer*]]

Displays details of the registered products on the system. If you do not supply any arguments, `lmf list` displays a one line summary of the PAK data for each product in the LDB.

The `full` argument displays the complete license details for each product. The *source* argument determines the source of the license information. There are three choices for *source*:

ldb	Displays a summary for each product in the LDB.
cache	Displays a summary for each product in the kernel cache. The kernel cache contains the license data used by the license checking functions.
all	Displays a combined summary for each product in the LDB, and for each product in the kernel cache.

history [*length*] [**from** *date*] [**for** *product* [*producer*]]

Lists data from the license management history file. The history file is maintained by the `lmf` utility and is a record of the LDB Maintenance Commands (`register`, `disable`, `enable`, `issue`, `cancel`, `delete`, `modify` and `amend`). The creation of a new LDB is also recorded in the history file. The history data is output with the most recent operations first. The data for each `lmf` command recorded comprises the product identity, the date and time the command was issued, and the fields that were changed on the license.

There are two choices for *length*:

short	Displays a one-line summary of the history data for each command issued.
full	Displays the history data for each command issued, and the license as it appeared before the command was issued.

The `from` *date* argument displays history data for commands issued after the *date* specified. The *date* argument can be specified in most common formats but the order must be: day, month, year. You do not need to use a separator between the day and month, or the month and the year. For example, 1st July 1989 could be specified as: 1-jul-1989, 1/7/89, 010789, or 1.july.89.

LDB Maintenance Commands

register [*input*]

Registers data from a PAK into the LDB. If you do not specify any arguments, the command displays a template which includes the fields that occur on the PAK. An editor is invoked so that you can add the license data to the appropriate fields. The editor used is defined by the EDITOR environment variable, but if this is not set, `/usr/ucb/vi` is used. When you leave the editor, the LMF scans the completed template to ensure that all the license data has been entered correctly. If it has not, an appropriate error message is displayed and you are given an opportunity to re-enter the editor and correct any mistakes.

When you have successfully registered a license you should store the PAK in a safe place; the PAK is a valuable proof of purchase and represents your license from Digital Equipment Corporation to use a software product.

There are two choices for *input*:

filename	Displays a copy of the file specified and invokes an editor so you can fill in any additional license details before registering the license data in the LDB.
–	Registers license data direct from standard input to the LDB.

Use the `lmf register` *filename* command to register license data from a file on your system that contains a partially complete PAK. When you have finished editing the license data, the LMF scans the file and gives you the opportunity to correct any mistakes.

Use the `lmf register –` command to register license data direct from standard input. You can also register a PAK from a file that already has valid license data, for example:

```
# lmf register - < filename
```

The command shown in the previous example does not display the contents of the file, neither does the command allow you to edit the file. However, the LMF does scan the file to ensure format and data is correct. If the license data is valid, the license is registered in the LDB, if not, the appropriate error message is displayed.

disable *product* [*producer* [*authorization*]]

Disables a license from use on the system. Licenses which have been disabled remain in the LDB, but do not get copied into the kernel cache. The `lmf disable` command does not have an immediate affect on the kernel cache. To remove a license from the kernel cache straight away, use the `lmf unload` command.

enable *product* [*producer* [*authorization*]]

Enables a license for use on the system. Licenses are automatically enabled when they are registered.

The `lmf enable` command does not have an immediate affect on the kernel cache. To load the license details into the kernel cache straight away, use the `lmf load` command.

issue *file product* [*producer* [*authorization*]]

Issues a reconstructed PAK for the product and removes the license from the LDB. The reconstructed PAK is output to the *file* given in the command. If the PAK is issued without errors, the license is automatically removed from the kernel cache. The format of the reconstructed PAK is suitable for registering using the `lmf register -` < *filename* command. You can use `lmf issue` to transfer a license from one system to another. The license is revoked on the executing system and a PAK is produced which can be registered on another system.

cancel *date product* [*producer* [*authorization*]]

Cancels the license on the *date* given. This allows you to stop use of the product earlier than the date shown by the Key Termination Date field on the PAK. You can change the cancellation date more than once; by reissuing the `lmf cancel` command with a different *date* argument.

The *date* argument can be specified in most common formats but the order must be: day, month, year. You do not need to use a separator between the day and month, or the month and the year.

The `lmf cancel` command does not have an immediate affect on the kernel cache. To update the license details for the product in the kernel cache straight away, use the `lmf load` command.

delete *product* [*producer* [*authorization*]]

Deletes a license from the LDB and the kernel cache.

Before you do this you should ensure that you have a record of the license in your files.

modify *product* [*producer* [*authorization*]]

Modifies the unprotected fields on a license. You can only modify the Comments field and, if the license has the MOD_UNITS Key Option, the Number of Units field. These fields have a colon (:) after the field name, and changes to fields other than Comments and Number of Units are ignored. The editor used is the same as for `lmf register`.

The command does not have an immediate effect on the kernel cache, so you should use the `lmf load` command to update the license for the product in the kernel cache.

amend *product* [*producer* [*authorization*]]

Amends the protected fields on a license. You must only use this command when you need to register the license data from a Product Authorization Amendment (PAAM). The fields you can change have a colon (:) after the field name. You must make all the changes shown on the PAAM, including entering a new Checksum. Changes made to fields without colons after the field name are ignored. The editor used is the same as for `lmf register`.

The command does not have an immediate effect on the kernel cache, so you should use the `lmf load` command to update the license for the product in the kernel cache.

Service Commands

reset [**cpus** [*n*]]

Rescans the LDB so that any changes that have been made are copied to the kernel cache. If you do not supply any arguments, `lmf reset` copies the license details for all products from the LDB to the kernel cache.

Use the `lmf reset cpus` command to copy license details from the LDB to the kernel cache, and to determine the System Marketing Model (SMM) by using the number of active CPUs. The SMM is the model name of a computer system, as used in marketing and pricing and is read by the LMF when the system is rebooted. The SMM is used by some products to define the number of license units needed in the kernel cache before access to the product is granted.

Use the `lmf reset cpus` *n* command to copy license details from the LDB to the kernel cache, and to determine the SMM by using *n* as the number of active CPUs.

load *users product* [*producer* [*authorization*]]

Loads enough license units into the kernel cache to enable use of the product by the number of *users* specified. The number of license units required for the specified number of *users* is calculated by the LMF. An appropriate number of units must be registered in the LDB before issuing the `load` command.

If you specify zero (0) as the *users* argument, all the license units for the product are loaded into the kernel cache. If the product is Availability Licensed, you must always specify zero (0) as the *users* argument.

unload *users product* [*producer*]

Unloads enough license units from the kernel cache to reduce the use of the product by the number of *users* specified. The number of license units required for the specified number of *users* is calculated by the LMF. After you have issued the command any existing users of the product are allowed to finish using it before the new limit is imposed.

If you specify zero (0) as the *users* argument, all the license units for the product are removed from the kernel cache. If the product is Availability Licensed, you must always specify zero (0) as the *users* argument.

See Also

lmfsetup(8)
Guide to Software Licensing

Name

lmfsetup – License Management Facility PAK registration script

Syntax

/etc/lmfsetup [*template*]

Description

The `lmfsetup` script allows you to register data supplied by a Product Authorization Key (PAK). The `lmfsetup` script prompts you for the data associated with each of the fields on a PAK. When all the data has been entered, the License Management Facility (LMF) ensures there are entries against all the mandatory fields, and that the Checksum validates all the license data. If the data has been entered correctly, the PAK is registered in the License Database. If the data has been entered incorrectly, the appropriate error message is displayed and you are given the opportunity to re-enter the data.

The *template* option allows you to register license data from templates in `/usr/var/adm/lmf`. A template containing a partially complete PAK is created by some products as part of their installation process. The script only prompts you for data on the fields that are empty in the template. If the script cannot find the specified template in `/usr/var/adm/lmf` it searches the current directory.

The `lmfsetup` script is provided as an alternative to the `lmf register` command. This displays a template, which includes the fields on the PAK, and invokes an editor so that you can add the license data to the appropriate field. The `lmf register` command also allows errors to be corrected without having to re-enter all the data.

See Also

lmf(8)
Guide to Software Licensing

Name

ln01of – LN01(S) laser printer filter

Syntax

/usr/lib/lpdfilters/ln01of [**–c**] [**–n***login*] [**–h***host*] [**–w***width*] [**–l***num*] [**–i***indent*]
[**accounting file**]

Description

The `ln01of` filter is used to filter text data destined for the LN01 and LN01S laser printers. The filter handles the device dependencies of the printers and performs accounting functions. Accounting records are written to the file specified by the **af** field in `/etc/printcap` at the time of completion for each job.

The filter can handle plain ASCII files and files that have been preprocessed by nroff. Since nroff only deals with monospaced characters this limits the number of fonts that will work effectively. The fonts and font selection files are in `/usr/lib/font/devln01.` The ln01of filter translates nroff control sequences for underlining into the correct LN01(S) control sequences, but cannot handle superscripting and subscripting. It will pass through any escape or control sequences.

The `ln01of` filter can be the specified filter in both the **of** and the **if** fields in the `/etc/printcap` file, see `printcap`(5). When both fields are specified the **of** filter is used only to print the banner page. It is then stopped to allow the **if** filter access to the printer. The **if** filter maintains accounting information.

If the **of** field is the only one specified the filter is used to print the banner page. It is then stopped and restarted. This allows the **of** filter to be used to maintain accounting information.

If the **if** field is the only one specified the banner page will be sent directly to the printer. If the printer has been left in an undetermined state the banner page may not print correctly.

The best results will be obtained when the filter is specified in both the **of** and **if** fields. For a more detailed discussion on filters see the ''Line Printer Spooler Manual'' in the *ULTRIX Supplementary Documents, Volume 2: Programmer.*

Options

The arguments passed to the filter depend on its use. The **of** filter is called with the following arguments:

ln01of –w*width* **–l***length*
The *width* and *length* values come from the **pw** and **pl** fields in the `/etc/printcap` database. The **if** (or restarted **of**) filter is passed the following arguments:

ln01of –c –n*login* **–h***host* **–w***width* **–l***num* **–i***indent* **accounting file**
The **–c** flag is optional, and supplied only when control characters are to be printed, that is, when the **–l** option of `lpr`(1) is used to print the file. The **–w** and **–l** arguments are the same as for the **of** filter, however, they may have different values if the **–w** and/or **–z** options of `lpr`(1) were used to print the file. The **–n** and **–h** arguments specify the login name and host name of the job owner. These arguments are used to record

accounting information. The **–i** option specifies the amount of indentation to be used. The last argument is the name of the accounting file specified from the **af** field in the `/etc/printcap` database.

Diagnostics

The **lf** field (default /dev/null) in the `/etc/printcap` database specifies error logging file name.

Files

`/etc/printcap`
Printer capabilities database

`/dev/lp?`

See Also

lpr(1), pr(1), printcap(5), lpd(8), MAKEDEV(8), pac(8)
''Line Printer Spooler Manual,'' *ULTRIX Supplementary Documents, Volume 2: Programmer*

Name

ln01pp – LN01(S) print filter

Syntax

/usr/lib/lpdfilters/ln01pp

Description

The `ln01pp` filter is the print filter for print function `pr`(1). When the **–p** option is specified in `lpr`(1) to print a job, the `ln01pp` filter is invoked. The filter sends the appropriate control sequences to the LN01(S) to set the size of the printable area on a sheet of paper. The filter then calls `/bin/pr`.

The `ln01pp` filter should be specified in the **pp** field of the `/etc/printcap` file. For further information, see `printcap`(5). It is invoked when the **–p** option is used with the `lpr`(1) command to print a job.

Files

`/etc/printcap`
Printer capabilities database

`/dev/lp?`

See Also

lpr(1), pr(1), printcap(5), MAKEDEV(8)
"Line Printer Spooler Manual," *ULTRIX Supplementary Documents, Volume 2: Programmer*

Name

ln03of – LN03(S) laser printer filter

Syntax

/usr/lib/lpdfilters/ln03of [**–c**] [**–n***login*] [**–h***host*] [**–w***width*] [**–l***num*] [**–i***indent*]
[**accounting file**]

Description

The `ln03of` filter is used to filter text data destined for the LN03 and LN03S laser printers. The filter handles the device dependencies of the printers and performs accounting functions. Accounting records are written to the file specified by the **af** field in `/etc/printcap` at the time of completion for each job.

The filter can handle plain ASCII files and files that have been preprocessed by nroff. The `ln03of` filter translates nroff control sequences for underlining, superscripting, and subscripting into the correct LN03(S) control sequences.

The `ln03of` filter can be the specified filter in both the **of** and the **if** fields in the `/etc/printcap` file. For further information, see `printcap`(5). When both fields are specified the **of** filter is used only to print the banner page. It is then stopped to allow the **if** filter access to the printer. The **if** filter maintains accounting information.

If the **of** field is the only one specified the filter is used to print the banner page. It is then stopped and restarted. This allows the **of** filter to be used to maintain accounting information.

If the **if** field is the only one specified the banner page will be sent directly to the printer. If the printer has been left in an undetermined state the banner page may not print correctly.

The best results will be obtained when the filter is specified in both the **of** and **if** fields. For a more detailed discussion on filters see the ''Line Printer Spooler Manual'' in the *ULTRIX Supplementary Documents, Volume 2: Programmer.*

Options

The arguments passed to the filter depend on its use. The **of** filter is called with the following arguments:

ln03of –w*width* **–l***length*
The *width* and *length* values come from the **pw** and **pl** fields in the `/etc/printcap` database. The **if** (or restarted **of**) filter is passed the following arguments:

ln03of –c –n*login* **–h***host* **–w***width* **–l***num* **–i***indent* **accounting file**
The **–c** flag is optional, and supplied only when control characters are to be printed, that is, when the **–l** option of `lpr`(1) is used to print the file. The **–w** and **–l** arguments are the same as for the **of** filter, however, they may have different values if the **–w** and/or **–z** options of `lpr`(1) were used to print the file. If the value of the width is greater than 80 the job will be printed in landscape mode with a slightly smaller font. The length of the page is assumed to be 66 regardless of the length specified. The **–n** and **–h** arguments specify the login name and host name of the job owner.

These arguments are used to record accounting information. The **–i** option specifies the amount of indentation to be used. The last argument is the name of the accounting file specified from the **af** field in the `/etc/printcap` database.

Diagnostics

The **lf** field (default /dev/null) in the `/etc/printcap` database specifies error logging file name.

Files

`/etc/printcap`
Printer capabilities database

`/dev/lp?`

See Also

lpr(1), pr(1), printcap(5), lpd(8), MAKEDEV(8), pac(8)
''Line Printer Spooler Manual,'' *ULTRIX Supplementary Documents, Volume 2: Programmer*

Name

ln03rof, ln03rof_isolatin1, ln03rof_decmcs – LN03R ASCII to PostScript (TM) translation filters

Syntax

/usr/lib/lpdfilters/ln03rof [**–c**] [**–n***login*] [**–h***host*] [**–w***width*] [**–l***num*] [**–i***indent*] [**accounting file**]

Description

The `ln03rof` filter translates ASCII to PostScript (TM), handles the device dependencies of a PostScript (TM) printer and performs some accounting functions. PostScript (TM) printers print documents formatted in the PostScript (TM) Page Description Language only.

The `ln03rof_isolatin1` and `ln03rof_decmcs` filters are alternatives to `ln03rof`, and affect how ASCII documents are printed. They cause the encoding vectors for ISO Latin1 or DEC MCS to be used instead of the Adobe Standard Encoding Vector described in the *LN03R PostScript Programmers Supplement*. The `ln03rof` filter uses idle cycles to cache the fonts when accessed by the standard encoding vector. The cached fonts cannot be used by `ln03rof_isolatin1` or `ln03rof_decmcs` which means that small jobs will be slowed down.

The `ln03rof` filter is transparent to documents already formatted in PostScript (TM). The filter assumes that all other documents are plain ASCII and translates them into PostScript (TM). The decision to translate is based on the first two characters in the document. If they are '`%!`', the filter assumes the document has already been formatted in PostScript (TM). If the first two characters are not '`%!`, the filter assumes that the document is plain ASCII and translates it into PostScript (TM). The `ln03rof` filter ensures that the printer receives documents formatted in PostScript (TM) only. The `ln03rof` filter maintains accounting information only for documents that it translates. You specify the accounting file with the **af** field in the `/etc/printcap` file. For further information, see `printcap`(5).

You can specify the `ln03rof` filter in both the **of** and the **if** fields of the `/etc/printcap` file. The specification of the filter in the `/etc/printcap` database determines how the `lpr`(1) command affects the printed document. For example:

- If you specify the **of** field only and do not suppress banner pages, the filter will *always* translate the document into PostScript (TM).
- If you specify the **if** field and do not suppress banner pages, the document will not print. In this case, the banner is sent directly to the printer. Since the banner is not formatted in PostScript (TM), the printer cannot print the document and aborts the job.
- If you suppress the banner by including the **sh** field in `/etc/printcap` or by printing the file using the `lpr`(1) with the **–h** option, the document prints correctly.
- You can obtain the best results by specifying the filter in both the **of** and **if** fields.

For a more detailed discussion on filters see the "Line Printer Spooler Manual" in the *ULTRIX Supplementary Documents, Volume 2: Programmer.*

Options

The arguments passed to the filter depend on its use. The **of** filter is called with the following arguments:

ln03rof **–w***width* **–l***length*
: The *width* and *length* values come from the **pw** and **pl** fields in the `/etc/printcap` database. The **if** filter is passed the following arguments:

ln03rof **–c** **–n***login* **–h***host* **–w***width* **–l***num* **–i***indent* **accounting file**
: The **–c** flag is optional. Use it to print control characters when you use the **–l** option of `lpr`(1) to print the file. The **–w** and **–l** arguments are the same as for the **of** filter, however, they may have different values if the **–w** and/or **–z** options of `lpr`(1) were used to print the file. If the value of the width is greater than 80 the job will be printed in landscape mode with a slightly smaller font. If the value of the length is greater than 88 a smaller font will be used. The **–n** and **–h** arguments specify the login name and host name of the job owner. These arguments are used to record accounting information. The **–i** option specifies the amount of indentation to be used. The last argument is the name of the accounting file specified from the **af** field in the `/etc/printcap` database. These arguments only affect documents that are translated from ASCII to PostScript (TM).

Diagnostics

The **lf** field (default /dev/null) in the `/etc/printcap` database specifies error logging file name.

Files

`/etc/printcap`
: Printer capabilities database

`/dev/lp?`

See Also

lpr(1), pr(1), printcap(5), lpd(8), MAKEDEV(8), pac(8)
"Line Printer Spooler Manual," *ULTRIX Supplementary Documents, Volume 2: Programmer*

Name

load – down-line load software to a target node

Syntax

/etc/load [*node*]

Description

The `load` command loads software to an unattended target node. The target node is loaded by the host that executes the `load` command.

The `load` command requires the identification of the service circuit over which the load is performed, the Ethernet hardware address of the target node, and the service password needed to gain access to the target. This information is included in the nodes database entry for the target node. A node entry is defined with the `addnode` command. For further information, see `addnode`(8). The *node* is the name or address of the target node. A node name consists of from one to six alphanumeric characters. For single networks, a node address consists of a decimal integer (1-1023). For multiple networks, a node address consist of two decimal integers (n.n), where the first indicates the network number (1-63), and the second indicates the node address (1-1023).

Options

–p Uses the specified service and password (next arguments) in accessing the target node. You can omit a target node's service and password from the nodes database for security reasons, but you must then specify the service and password in the command line by using the `-p` option.

Examples

```
# /etc/load bangor -p 130fe
```

This command causes node `bangor` to be loaded by the ULTRIX host node executing the command. The ULTRIX host uses the load files specified in the nodes database entry for node `bangor` to perform the load.

See Also

addnode(8), ccr(8), getnode(8), mop_mom(8), remnode(8), trigger(8)
Guide to Local Area Transport Servers

Name

lockd – network lock daemon

Syntax

/usr/etc/lockd [**–t** *timeout*] [**–g** *graceperiod*]

Description

The `lockd` daemon processes lock requests that are either sent locally by the kernel or remotely by another lock daemon. The NFS locking service makes this advisory locking support possible by using the `fcntl` system call and the `lockf` subroutine. The `lockd` daemon forwards lock requests for remote data to the server site's lock daemon. The `lockd` daemon then requests the status monitor daemon, `statd,` for monitor service. The reply to the lock request is not sent to the kernel until the status daemon and the server site's lock daemon have replied.

If either the status monitor or server site's lock daemon is unavailable, the reply to a lock request for remote data is delayed until all daemons become available.

When a server recovers, it waits for a grace period for all client site `lockd` daemons to submit reclaim requests. Client site `lockd` daemons are notified by `statd` of the server recovery and promptly resubmit previously granted lock requests. If a client site's `lockd` daemon fails to secure previously granted locks at the server site, the `lockd` daemon sends the signal SIGLOST to all the processes that were previously holding locks and cannot reclaim them.

Options

–t *timeout*	The `lockd` daemon uses *timeout* (in seconds) as the interval instead of the default value of 15 seconds to retransmit a lock request to the remote server.
–g *graceperiod*	The `lockd` daemon uses *graceperiod* (in seconds) as the grace period duration instead of the default value of 45 seconds.

See Also

fcntl(2), lockf(3), signal(3), statd(8c)

Name

lpc – line printer control program

Syntax

/etc/lpc [*command* [*argument...*]]

Description

The `lpc` program is used by the system administrator to control the operation of the line printer system. For each line printer with an entry in the `/etc/printcap` file, `lpc` may be used to start/stop several different functions.

Without any arguments, `lpc` prompts for commands from the standard input. If arguments are supplied, `lpc` interprets the first argument as a command and the remaining arguments as parameters to the command. The standard input may be redirected causing `lpc` to read commands from file. The commands and their parameters are shown below. Commands may be abbreviated.

? [*command...*]
help [*command...*]

Prints a short description of each command specified in the argument list, or, if no arguments are given, a list of the recognized commands.

abort [*all*] [*printer...*]

Terminates an active spooling daemon on the local host immediately and then disables printing (preventing new daemons from being started by `lpr` for the specified printers.

clean [*all*] [*printer...*]

Removes all files beginning with ''cf'', ''tf'', or ''df'' from the specified printer queue(s) on the local machine.

enable [*all*] [*printer...*]

Enables spooling on the local queue for the listed printers. This will allow `lpr` to put new jobs in the spool queue.

exit
quit

Exits from `lpc`.

disablefR [*all*] [*printer...*]

Turns off the specified printer queues. This prevents new printer jobs from being entered into the queue by `lpr`.

restart [*all*] [*printer...*]

Attempts to start a new printer daemon. This is useful when some abnormal condition causes the daemon to die unexpectedly leaving jobs in the queue. The `lpq` command will report that there is no daemon present when this condition occurs.

start [*all*] [*printer...*]

Enables printing and starts a spooling daemon for the listed printers. This command also creates an `init` file in the printer's spool directory. The `init` file can be used by filters as an initialization flag.

status [*all*] [*printer...*]

Displays the status of daemons and queues on the local machine.

stop [*all*] [*printer...*]

Stops a spooling daemon after the current job completes and disables printing.

topq printer [*jobnum...*] [*user...*]

Places the jobs in the order listed at the top of the printer queue.

Diagnostics

?Ambiguous command
Abbreviation matches more than on command.

?Invalid command
No match was found.

?Privileged command
The command can only be executed by root.

Files

`/etc/printcap`	Printer description file
`/usr/spool/*`	Spool directories
`/usr/spool/*/lock`	Lock file for queue control

See Also

lpq(1), lpr(1), lprm(1), printcap(5), lpd(8)

Name

lpd – line printer daemon

Syntax

/usr/lib/lpd [**–l**] [**–L** *logfile*] [*portnumber*]

Description

The `lpd` line printer daemon uses the system calls `listen` and `accept`, to receive requests to print files in the print queue, transfer files to the spooling area, display the queue, and remove jobs from the queue.

The line printer daemon is invoked by the `/etc/rc` command file when the system goes multi-user (normally at system start up). The daemon looks at the `/etc/printcap` file to find out about the capabilities of existing printers, and prints any files that were not printed when the system last stopped operating.

The Internet port number used to rendezvous with other processes is normally obtained with `getservbyname`, but can be changed by using the *portnumber* argument.

Access to the facilities provided by the `lpd` daemon is controlled by only allowing requests from the machines listed in the `/etc/hosts.equiv` or `/etc/hosts.lpd` files. The `/etc/hosts.equiv` file is described on the `hosts.equiv(5yp)` reference page. The `/etc/hosts.lpd` file is a list of names consisting of one host machine name per line. An * character at the start of any line in `/etc/hosts.lpd` allows print requests from all systems. The machine names listed in the `/etc/hosts.equiv` and `/etc/hosts.lpd` files may optionally contain the local BIND domain name. For more information on BIND, see the *Guide to the BIND/Hesiod Service*.

You can also control access to the `lpd` daemon by specifying the **rs** capability for a particular printer in the `printcap` file. This restricts the printer users to those with accounts on the machine which the printer is connected to.

The file *lock* in each spool directory is used to prevent multiple daemons from becoming active simultaneously, and to store information about the daemon process for `lpr`, `lpq`, and `lprm`.

After the daemon has successfully set the lock, it scans the directory for command files with names beginning with *cf*. These files specify names of files which are to be printed and parameters affecting how the files are printed. Each line in a command file begins with a key character to specify what to do with the remainder of the line. The key characters and their meanings are shown below. They are listed in the order that they would appear in a command file.

If a file is to be printed but can not be opened, a message will be placed in the *logfile* (by default, the system console).

The `lpd` daemon uses `flock` to provide exclusive access to the *lock* file. The lock is automatically removed by the kernel when a `lpd` process terminates for any reason. The lock file contains two lines. The first is the process id of the daemon and the second is the control file name of the current job being printed. The second line is updated for use by `lpq` and `lprm`.

The key characters in the command file and their meanings are as follows:

H*host*	Host Name. Name of the machine where `lpr` was invoked.
P*user*	Person. Login name of the person who invoked `lpr`. This is used to verify ownership by `lprm`.
J*job*	Job Name. String to be used for the job name on the banner page.
C*class*	Classification. String to be used for the classification line on the banner page.
L*user*	Literal. The line contains identification information from the password file and causes the banner page to be printed.
I*num*	Indent. The number of characters by which the output is indented (in ASCII).
M*user*	Send mail to the specified user when the current print job completes.
1*font*	Troff Font R. Name of the font file to use instead of the default.
2*font*	Troff Font I. Name of the font file to use instead of the default.
3*font*	Troff Font B. Name of the font file to use instead of the default.
4*font*	Troff Font S. Name of the font file to use instead of the default.
W*num*	Width. Changes the page width (in characters) used by `pr` and the text filters.
Z*num*	Length. Changes the page length (in lines) used by `pr` and the text filters.
D*data_type*	Data type of the job. (PostScript (TM) printers only.)
<*input_tray*	Selects the input tray that supplies paper for the print job. (PostScript (TM) printers only.)
>*output_tray*	Selects the output tray where the printed paper is deposited. (PostScript (TM) printers only.)
O*orientation*	Specifies the orientation of the printed output on the page. (PostScript (TM) printers only.)
F*page_size*	Specifies the size of the pages being printed. (PostScript (TM) printers only.)
S*sheet_size*	Specifies the physical size of the sheets being printed. (PostScript (TM) printers only.)
E*message*	Specifies what happens to messages generated when processing the print job. (PostScript (TM) printers only.)
X*num*	Specifies the number of times each page is printed. (PostScript (TM) printers only.)
A*num*	Specifies the first page to be printed for the job. (PostScript (TM) printers only.)
B*num*	Specifies the last page to be printed for the job. (PostScript (TM) printers only.)

G*num*	Specifies the number of pages to be printed on a single physical sheet. (PostScript (TM) printers only.)
z*filename*	Layup definition file which alters the appearance of pages (margins, borders, etc). (PostScript (TM) printers only.)
K*sides*	Specifies whether the job should be printed on both sides to the physical sheet, and whether the pages should be rotated by 180 degrees. (PostScript (TM) printers only.)
T*title*	Title. String to be used as the title for `pr`.
f*filename*	Formatted File. Name of a file to print which is already formatted.
p*filename*	Name of a file to print using `pr` as a filter.
l*filename*	Like `f` but passes control characters and does not make page breaks.
t*filename*	Troff File. The file contains `troff`.
n*filename*	Ditroff File. The file contains `ditroff` output (device independent `troff`).
d*filename*	DVI File. The file contains TeX output (DVI format from Stanford).
g*filename*	Graph File. The file contains data produced by `plot`.
v*filename*	The file contains a raster image.
c*filename*	Cifplot File. The file contains data produced by cifplot.
r*filename*	The file contains text data with Fortran carriage control characters.
x*filename*	Do not interpret any control characters in the file.
U*filename*	Unlink. Name of file to remove upon completion of printing.
N*name*	File name. The name of the file which is being printed, or a blank for the standard input (when `lpr` is invoked in a pipeline).

PostScript (TM) Printers With Specialised Support

To use the features of the PostScript (TM) printers for which specialised support is available, you must ensure that **:ps=LPS:** is set in the appropriate entry in the `/etc/printcap` file.

For PostScript (TM) printers, the line printer daemon assembles the PostScript (TM) from the users data files and PostScript (TM) device control modules. The device control modules access device features and manipulate the appropriate printer parameters. Device control modules (the **Dl** capability in `/etc/printcap`), are provided in an archive file, refer to `ar(5)`. Device control modules access printer specific features of PostScript (TM) and are therefore device dependent.

The data type of the spooled files is given by the **Da** capability in the `printcap` file. The data type can be overridden by the **D** key character in the command file.

The support for PostScript (TM) printers introduces a new method of specifying the type of the data to be printed using the **–D** option of `lpr` or the **Da** capability in `/etc/printcap`. The mechanism for invoking these translators needs to interact with the existing mechanism for invoking filters for non-text files, so that the existing mechanism can be used if required. The mechanism has the following features:

> The new translators are invoked via the shell program `xlator_call`, which is passed the data type as one of its parameters.
>
> Any old style filter arguments (for example, **–t** and **–x**) take precedence over **–D** (even though `lpr` will accept the combination). This means that the **if** capability in `/etc/printcap` must not be specified if data types are being used.

Filter Capabilities

Two of the `printcap` capabilities that affect the behaviour of `lpd` require a more detailed explanation than that given on the `printcap(5)` reference page.

ct Connection type. This entry in the `printcap` file determines the type of connection. The following table shows the valid choices for **ct**, and the mandatory and optional entries to go with each choice.

ct=	Mandatory Entries	Optional Entries
dev	lp	of
lat	lp, ts, op, os	of
remote	rp, rm	
network	of	

Remote means a printer that is connected to another system running a compatible printing daemon. Network means that the output filter does not use stdout (that is, no **lp**) and is restarted for each job.

uv ULTRIX version. To enable the **ct** capability to determine the type of connection, and to ensure % escapes are expanded in all filter command strings, you must have

```
:uv=psv1.0:
```

as part of the `printcap` entry.

Using Filter Capabilities

This section refers to the programs specified by the following filter capabilities in the `printcap` file: `cf`, `df`, `gf`, `if`, `nf`, `of`, `pp`, `tf`, `vf`, `xf`, `Lf`, and `Xf`.

Filters may be specified as pipelines as well as simple commands. The syntax accepts tab or space as word separators, and | to set up a pipe connection. You can specify arguments to the filters. The individual commands may be specified as full path names or as simple program names, in which case the path searched is:

```
/usr/local/lib/lpdfilters:/usr/ucb:/bin:/usr/bin:
                                /usr/lib:/usr/lib/lpdfilters
```

Filter Argument Parameterisation

The arguments to the filter program are normally supplied automatically by `lpd` but you can assign your own arguments to filters. When arguments are supplied automatically, they are put after the arguments for the first command in a pipeline, or after the arguments of a simple command.

If you want to assign your own arguments to filters, you must ensure that **:uv=psv1.0:** is set in the appropriate entry in the `printcap` file.

Arguments are supplied exactly as for previous `lpd` releases unless a special character (%) is found anywhere in the command. If a % is found it tells `lpd` that you have taken control of passing arguments to the filter. In this case, the default set of arguments are no longer appended to the argument list. The arguments supplied to the command are those specified in the command string, but with `lpd` replacing %x pairs with parameter strings.

To allow the special character, %, to be passed, a pair of %%'s are replaced by a %. It is an error to specify an unknown %x pair. The %x pairs and the parameter strings that replace them are shown below.

%x Pairs	Parameter String
%%	%
%0	null string (used to pass null argument or disable default arguments)
%_	space (used to insert spaces in arguments)
%A	accounting file
%D	data type
%F	pagesize
%H	host where job was submitted
%I	indent
%j	job name
%J	job id as shown by `lpq`
%L	length
%O	orientation
%P	printer name
%T	title
%U	user's login name
%W	width
%X	pixel width
%Y	pixel length

All of the above %x pairs are available to all input filters. For a network printer (**:ct=network:** in the `/etc/printcap` file) they are all available to the output filter **of**. For dev and lat printers the per-job parameters (%H, %U, %J and %j) are not available to the output filter. This is because the output filter is invoked per session so that per-job parameters are still unset.

Filter Arguments Supplied By lpd

The arguments automatically assigned to each filter by `lpd` are shown below. For clarity the parameters for each argument are shown as the appropriate %x pair.

Filter	Parameterised Form of Default Arguments
cf	-x%X -y%Y -n %U -h %H %A
df	-x%X -y%Y -n %U -h %H %A
gf	-x%X -y%Y -n %U -h %H %A
if (when `lpr` is invoked without the **–l** option)	-w%W -l%L -n %U -h %H %A
if (when `lpr` is invoked with the **–l** option)	-c -w%W -l%L -n %U -h %H %A
nf	-w%W -l%L -n %U -h %H %A
of	-w%W -l%L
pp	-w%W -l%L -h %T
rf	-w%W -l%L -n %U -h %H %A
tf	-x%X -y%Y -n %U -h %H %A
vf	-x%X -y%Y -n %U -h %H %A
xf	(no arguments)
Lf	(no arguments)
Xf	%D %O %F %W %L %I

The %A argument is only supplied if the **af** capability is present.

Options

–l Log valid requests received from the network. This can be useful for debugging purposes.

–L*logfile*
Write error conditions to the file specified by the argument *logfile*. If this option is not used, error conditions are written to the system console.

Files

`/etc/printcap`	Printer description file
`/usr/spool/lpd`	Spool directories
`/dev/lp*`	Line printer devices
`/dev/printer`	Socket for local requests
`/etc/hosts.lpd`	Lists machine names allowed printer access
`/etc/hosts.equiv`	Lists machine names allowed printer access as trusted machines

See Also

lpq(1), lpr(1), lprm(1), printcap(5), lpc(8), pac(8), xlator_call(8)
Guide to the BIND/Hesiod Service
Guide to System Environment Setup

Name

lpf – general purpose line printer filter

Syntax

/usr/lib/lpdfilters/lpf [**–c**] [**–n***login*] [**–h***host*] [**–w***width*] [**–l***num*] [**–i***indent*] [**accounting file**]

Description

The `lpf` filter handles text data destined for impact printers: LP25, LP26, LP27, LA50, LA75, LA100, LA120, LA210, LG01. The filter regulates the device dependencies of the printers and performs accounting functions. When the print job is done, `lpf` writes accounting records to the file specified by the **af** field in /etc/printcap.

The filter can handle plain ASCII files as well as files that have been preprocessed by nroff. However, it ignores escape sequences requesting superscripting and subscripting.

You can specify the `lpf` filter in both the **of** and the **if** fields of the `/etc/printcap` file. For further information, see `printcap`(5). When you specify both fields, the **of** filter prints the banner page only, then stops. The **if** filter then gains access to the printer. The **if** filter maintains accounting information.

If you specify the **of** field only, it prints the banner page then stops and restarts to maintain accounting information.

If you specify the **if** field only, it sends the banner page directly to the printer. This is not a problem for most impact printers.

For a more detailed discussion on filters see the ''Line Printer Spooler Manual'' in the *ULTRIX Supplementary Documents, Volume 2: Programmer.*

The arguments passed to the filter depend on its use. The **of** filter is called with the following arguments:

lpf –w*width* **–l***length*
: The *width* and *length* values come from the **pw** and **pl** fields in the `/etc/printcap` database. The **if** (or restart **of**) filter is passed the following arguments:

lpf –c –n*login* **–h***host* **–w***width* **–l***num* **–i***indent* **accounting file**

The **–c** flag is optional and supplied only when control characters are to be printed; when the **–l** option of `lpr`(1) is used to print the file. The **–w** and **–l** arguments are the same as for the **of** filter, however, they may have different values if the **–w** and/or **–z** options of `lpr`(1) were used to print the file. The **–n** and **–h** arguments specify the login name and host name of the job owner. These arguments are used to record accounting information. The **–i** option specifies the amount of indentation to be used. The last argument is the name of the accounting file specified from the **af** field in the `/etc/printcap` database.

Diagnostics

The **lf** field (default /dev/null) in the `/etc/printcap` database specifies error logging file name.

Files

`/etc/printcap`
Printer capabilities database

`/dev/lp?`

See Also

lpr(1), pr(1), printcap(5), lpd(8), MAKEDEV(8), pac(8)
''Line Printer Spooler Manual,'' *ULTRIX Supplementary Documents, Volume 2: Programmer*

Name

lprsetup – line printer spooler set up program

Syntax

/etc/lprsetup

Description

The `lprsetup` command provides an interactive easy-to-use facility for administrating the line printers on your system. The `lprsetup` program contains on-line help and default answers to questions about adding, deleting, or changing the characteristics of any of the line printers on your system. Whenever a question is asked, the default selection is given in [ ] . You can press Return in response to the question to accept the default, or enter an alternate value for the given parameter.

The program knows about all of the possible symbols in the `/etc/printcap` file. See `printcap`(5) for a current list. After you have entered a printer specification, and have verified that it is correct, `lprsetup` then creates the spooling directory, links the output filter, and creates an `/etc/printcap` entry for the new printer.

If the printer is connected to the local processor, however, you must specify the printer device name which will be in the **lp** printcap entry.

See Also

printcap(5)

Name

lpx – line printer exerciser

Syntax

/usr/field/lpx [**–h**] [**–o***file*] [**–p***n*] [**–t***m*] **–d***dev*

Description

The line printer exerciser outputs a rolling character pattern to the printer. Five pages are output and then the printer will pause for 15 minutes (default). Then 5 more pages are output, followed by a pause. This pattern continues until the process receives a <CTRL/C> or a kill -15 *pid.*

Disable the line printer queue of the printer to be tested before running `lpx`. Check the `/etc/printcap` file to determine the line printer queue, and then run line printer control program `/etc/lpc` to disable the printer. This will stop other jobs from interfering with the testing.

Arguments

You must specify the following function flag and its argument to the `lpx` exerciser:

–d*dev* The line printer device name and unit number to test as listed in the `/dev` directory. For example, lp, lp1.

Options

The `lpx` options are:

–h Prints the help messages for the `lpx` command.

–o*file* Saves the output diagnostics in *file*.

–p*n* Sets pause for *n* minutes. During the pause period, `lpx` will only exercise the controller, saving paper. The default value for *n* is 15. A value of *n* = 0 indicates no pause.

–t*m* Specifies the run time in minutes (*m*). The default is to run `lpx` until the process receives a <CTRL/C> or kill -15 *pid.*

Restrictions

If there is a need to run a system exerciser over an NFS link or on a diskless system there are some restrictions. For exercisers that need to write into a file system, such as `fsx`(8), the target file system must be writable by root. Also the directory, in which any of the exercisers are executed, must be writable by root because temporary files are written into the current directory. These latter restrictions are sometimes difficult to overcome because often NFS file systems are mounted in a way that prevents root from writing into them. Some of the restrictions may be overcome by copying the exerciser to another directory and then executing it.

Examples

The following example causes `lpx` to exercise lp1 until the process receives a <CTRL/C> or kill -15 *pid*.

```
% /usr/field/lpx -dlp1
```

The following example exercises lp for 120 minutes in the background.

```
% /usr/field/lpx -t120 -dlp &
```

See Also

Guide to System Exercisers

makedbm (8yp)

Name

makedbm – make a yellow pages dbm file

Syntax

makedbm [**–i** *yp_input_file*] [**–o** *yp_output_name*] [**–d** *yp_domain_name*] [**–m** *yp_master_name*] *infile outfile*
makedbm [**–u** *dbmfilename*]

Description

The makedbm command takes the file specified by the argument *infile* and converts it to a pair of files in dbm(3x) format, namely outfile.pag and outfile.dir. Each line of the input file is converted to a single dbm record. All characters up to the first tab or space form the key, and the rest of the line is defined as the key's associated data. If a line ends with a backslash (\), the data for that record is continued onto the next line. It is left for the clients of the yellow pages to interpret the number sign (#); makedbm does not treat it as a comment character. The *infile* parameter can be a hyphen (-), in which case makedbm reads the standard input.

The makedbm command is meant to be used in generating dbm files for the yellow pages service. The makedbm command generates a special entry with the key yp_last_modified, which is the date of *infile*.

Options

–i	Create a special entry with the key yp_input_file.
–o	Create a special entry with the key yp_output_name.
–d	Create a special entry with the key yp_domain_name.
–m	Create a special entry with the key yp_master_name. If no master host name is specified, yp_master_name will be set to the local host name.
–u	Undo a dbm file. That is, print out a dbm file one entry per line, with a single space separating keys from values.

Examples

The following example shows how a combination of commands can be used to make the yellow pages dbm files passwd.byname.pag and passwd.byname.dir from the /etc/passwd file. The percent sign (%) signifies the system prompt.

```
% awk 'BEGIN { FS = ":"; OFS = ""; }
 { print $1, $0 }' /etc/passwd > ptmp
% makedbm ptmp passwd.byname
% rm ptmp
```

The awk command creates the file *ptmp* which is in a form usable by makedbm. The makedbm command uses the *ptmp* file to create the yellow pages dbm files passwd.byname.dir and passwd.byname.pag. The rm command removes the *ptmp* file.

See Also

yppasswd(1yp), dbm(3x), ypmake(8yp)

Name

MAKEDEV – makes system special files

Syntax

/dev/MAKEDEV *device-name*?...

Description

The `MAKEDEV` shell script is normally used to install special files. It resides in the **/dev** directory, the normal location of special files. Arguments to `MAKEDEV` are usually of the form *device-name*? where *device-name* is one of the supported devices listed in Section 4 of the *ULTRIX Reference Pages* and ? is a logical unit number. A few special arguments create assorted collections of devices and are listed below.

DECstation*	Creates a DECstation specific setup.
std	Creates all *standard* devices for all systems.
local	Creates those devices specific to the local site. This request causes the shell file `/dev/MAKEDEV.local` to be executed. Site specific commands, such as those used to setup dialup lines as ttyd?, should be included in this file.

Because all devices are created using `mknod`(8), this shell script is useful only to the superuser.

Diagnostics

Either self-explanatory, or generated by one of the programs called from the script. Use `sh -x MAKEDEV` in case of trouble.

See Also

intro(4), config(8), mknod(8)

Name

MAKEDEV – makes system special files

Syntax

/dev/MAKEDEV *device-name?...*

Description

The `MAKEDEV` shell script is normally used to install special files. It resides in the `/dev` directory, the normal location of special files. Arguments to `MAKEDEV` are usually of the form *device-name?* where *device-name* is one of the supported devices listed in Section 4 of the *ULTRIX Reference Pages* and ? is a logical unit number. A few special arguments create assorted collections of devices and are listed below.

boot* Creates all *boot** and standard devices for a specific cpu type, for example, "boot8800".

mvax* Creates MicroVAX specific devices. This is superseded by the boot argument.

VAXstation*
Creates a VAXstation 2200 specific setup. This is superseded by the boot argument.

std Creates all *standard* devices for all systems.

local Creates those devices specific to the local site. This request causes the shell file `/dev/MAKEDEV.local` to be executed. Site specific commands, such as those used to setup dialup lines as ttyd?, should be included in this file.

Since all devices are created using `mknod`(8), this shell script is useful only to the superuser.

Diagnostics

Either self-explanatory, or generated by one of the programs called from the script. Use `sh -x MAKEDEV` in case of trouble.

See Also

intro(4), config(8), mknod(8)

Name

MAKEHOSTS – make symbolic links to hosts

Syntax

/usr/hosts/MAKEHOSTS

Description

The `MAKEHOSTS` command creates symbolic links to the `rsh` command for each of the hosts as defined in the `/etc/svc.conf` file. If the `/etc/svc.conf` file does not exist, then the `MAKEHOSTS` command creates the links for each of the hosts listed in the local `/etc/hosts` file. The resulting symbolic links populate the `/usr/hosts` directory.

The symbolic links are a convenience that allows users to type the name of a remote host on the network to log into that host. For example, the following command logs you into a remote host called `chicago`:

```
# chicago
```

If you do not create the symbolic links, you must use the `rlogin` command to log into a remote host. For example, to log into a remote host named `chicago`, type:

```
# rlogin chicago
```

See Also

rlogin(1c), rsh(1c), hosts(5), svc.conf(5)

Name

makekey – generate encryption key

Syntax

/usr/lib/makekey

Description

The `makekey` command improves the usefulness of encryption schemes depending on a key by increasing the amount of time required to search the key space. It reads 10 bytes from its standard input, and writes 13 bytes on its standard output. The output depends on the input in a way intended to be difficult to compute (that is, to require a substantial fraction of a second).

The first eight input bytes (the *input key*) can be arbitrary ASCII characters. The last two (the *salt*) are best chosen from the set of digits, uppercase and lowercase letters, the period (.), and the slash (/). The salt characters are repeated as the first two characters of the output. The remaining 11 output characters are chosen from the same set as the salt and constitute the *output key.*

The transformation performed is essentially the following: the salt is used to select one of 4096 cryptographic machines all based on the National Bureau of Standards DES algorithm, but modified in 4096 different ways. Using the input key as key, a constant string is fed into the machine and recirculated a number of times. The 64 bits that come out are distributed into the 66 useful key bits in the result.

The `makekey` command is intended for programs that perform encryption (for instance, `ed(1)`. Usually input and output of `makekey` will be pipes.

See Also

ed(1)

Name

memx – memory exerciser

Syntax

/usr/field/memx [**–h**] [**–s**] [**–o***file*] [**–t***i*] [**–m***j*] [**–p***k*]

Description

The memx memory exerciser spawns processes to exercise memory by writing and reading three patterns: 1's and 0's, 0's and 1's, and a random pattern.

You specify the number of processes to spawn and the size of memory to be tested by each process. The first process is a shared memory exerciser, the remaining are standard memory exercisers. The memx exerciser will run until the process receives a <CTRL/C> or a kill -15 *pid.*

A logfile is made in /usr/field for you to examine and then remove. If there are errors in the logfile, check the /usr/adm/syserr/syserr.<hostname> file, where the driver and kernel error messages are saved.

Options

The memx options are:

–h	Print the help message for the memx command.
–s	Disable shared memory testing.
–o*file*	Save diagnostic output in *file.*
–t*i*	Run time in minutes (*i*). The default is to run until the process receives a <CTRL/C> or a kill -15 *pid.*
–m*j*	The memory size in bytes (*j*) to be tested by each spawned process. Must be greater than 4095. The default is (total-memory)/20.
–p*k*	The number of processes to spawn (*k*). The default is 20. The maximum is also 20.

Restrictions

The memx exerciser is restricted by the size of swap space available. The size of the swap space and the size of internal memory available will determine how many processes can run on the system. For example, If there were 16Mbytes of swap space and 16Mbytes of memory, all of the swap space would be used if all 20 spawned memory exercisers were running. In that event, no new processes would be able to run. On systems with large amounts of memory and small swap space, you must restrict the number of memory exercisers and/or the size of memory being tested.

If there is a need to run a system exerciser over an NFS link or on a diskless system there are some restrictions. For exercisers that need to write into a file system, such as fsx(8), the target file system must be writable by root. Also the directory, in which any of the exercisers are executed, must be writable by root because temporary files are written into the current directory. These latter restrictions are sometimes difficult to overcome because often NFS file systems are mounted in a way that prevents root from writing into them. Some of the restrictions may be overcome by copying the exerciser to another directory and then executing it.

Examples

The following example tests all of memory by running 20 spawned processes until a <CTRL/C> or kill -15 *pid* is received.

```
% /usr/field/memx
```

The following example runs 10 spawned processes, memory size 500,000 bytes, for 180 minutes in the background.

```
% /usr/field/memx -t180 -m500000 -p10 &
```

See Also

Guide to System Exercisers

miscd (8c)

Name

miscd – miscellaneous services daemon

Syntax

/etc/miscd

Description

The `miscd` daemon is the service daemon for some utility internet services. It is started by `inetd`(8c) when it receives a packet or a connection on an appropriate socket. Once invoked, `miscd` determines the type of request and the protocol and then attempts to service it.

When performing services for datagram functions, the daemon will stay active until approximately two minutes pass without a service request. It will then terminate until reinvoked by `inetd`(8c.)

Invocations of `miscd` serving for connection-based sockets will terminate when the connection is broken.

Restrictions

The `miscd` daemon supports only a subset of the internet utility functions. The supported services are echo, discard, systat, daytime, quote, chargen, and time.

See Also

inetd(8c)

Name

mkconsole – a program to build or update boot console command files

Syntax

/etc/mkconsole [*kernel_image_name*]

Description

The `mkconsole` shell script is used to either build or update console storage media, as necessary, depending on the processor type. It uses the `sizer`(8) program to sense the processor type and accordingly, builds two boot command procedures required to boot the currently booted system disk. The VAX 11/780, VAX 11/785, VAX 11/730, VAX 8600 and VAX 8650 are the only supported processor types. After the console media is built or updated, you can boot the system using the **b** or **b ask** commands from the console mode prompt.

If a *kernel_image_name* is given as an argument, that file will be treated as the running kernel file and opened by the sizer program. The default kernel image name is /vmunix.

Files

```
/usr/sys/730cons/askboo.cmd
/usr/sys/780cons/askboo.cmd
/usr/sys/8600cons/askboo.com
/usr/sys/730cons/defboo.cmd
/usr/sys/780cons/defboo.cmd
/usr/sys/8600cons/defboo.com
```

See Also

sizer(8)
Guide to Shutdown and Startup

Name

mkfs – construct a file system

Syntax

/etc/mkfs [*-N*] *special size* [*nsect*] [*ntrack*] [*blksize*]
[*fragsize*] [*ncpg*] [*minfree*] [*rps*] [*nbpi*] [*opt*]

Description

File systems are normally created with the `newfs`(8) command.

The *-N* option is used to run `mkfs` in no update mode. In this mode, `mkfs` will not write to `special`.

The `mkfs` command constructs a file system by writing on the special file *special.* The numeric size specifies the number of sectors in the file system. The `mkfs` command builds a file system with a root directory and a *lost+found* directory. For further information, see `fsck`(8). The number of i-nodes is calculated as a function of the file system size. No boot program is initialized by `mkfs`. For further information, see `newfs`(8).

When the on-disks inodes of the file system are written, each contains a unique number in its generation number field. This number uniquely identifies each inode in a file system.

The optional arguments allow fine tune control over the parameters of the file system. The *nsect* argument specifies the number of sectors per track on the disk. The *ntrack* argument specifies the number of tracks per cylinder on the disk. The *blksize* argument gives the primary block size for files on the file system. It must be a power of two, currently selected from 4096 or 8192. The *fragsize* argument gives the fragment size for files on the file system. The *fragsize* argument represents the smallest amount of disk space that will be allocated to a file. It must be a power of two currently selected from the range 512 to 8192. The *ncpg* argument specifies the number of disk cylinders per cylinder group. This number must be in the range 1 to 32. The *minfree* argument specifies the minimum percentage of free disk space allowed. Once the file system capacity reaches this threshold, only the superuser is allowed to allocate disk blocks. The default value is 10%. If a disk does not revolve at 60 revolutions per second, the *rps* parameter may be specified. Users with special demands for their file systems are referred to ''A Fast File System for UNIX'' in the *ULTRIX Supplementary Documents, Volume 3: System Manager* for a discussion of the tradeoffs in using different configurations. The *nbpi* argument specifies the number (ratio) of bytes per inode. The default is 2048 bytes. The *opt* argument is used to indicate the whether the file system should optimize for space or time. The *opt* argument can be assigned a value of s or t.

See Also

dir(5), fs(5), fsck(8), newfs(8), tunefs(8)
''A Fast File System for UNIX'' *ULTRIX Supplementary Documents, Volume 3: System Manager*

Name

mklost+found – make a lost+found directory for fsck

Syntax

/etc/mklost+found

Description

A `lost+found` directory is created in the current directory and a number of empty files are created therein and then removed so that there will be empty slots for `fsck`(8). This command should not normally be needed since `mkfs`(8) automatically creates the `lost+found` directory when a new file system is created.

See Also

fsck(8), mkfs(8)

Name

mknod – make special file

Syntax

/etc/mknod name [**c**] [**b**] major minor
/etc/mknod name **p**

Description

The `mknod` command makes a special file. The first argument is the *name* of the entry. The second is **b** if the special file is block-type (disks, tape) or **c** if it is character-type (other devices). The last two arguments are numbers specifying the *major* device type and the *minor* device (for example, unit, drive, or line number).

The assignment of major device numbers is specific to each system. These numbers are obtained from the system source file *conf.c.*

The `mknod` command can also be used to create fifo's, also known as named pipes, (second case in the Syntax section above).

See Also

mknod(2)

Name

mkpasswd – generate hashed password table

Syntax

/etc/mkpasswd [–uv] *passwdfile*

Description

The `mkpasswd` command is used to generate the hashed password database used by the library routines `getpwnam()` and `getpwuid()`.

If the **–u** option is present the hashed files will not be built unless they already exist. In any case, the hashed files will not be built if the time of last modification of the *passwdfile* is less recent than that of an existing hashed files.

If the **–v** option is supplied, each entry will be listed as it is added.

The file *passwdfile* is usually `/etc/passwd` and must be in the format of `passwd(5)`. The `mkpasswd` command will generate database files named *passwdfile*`.pag` and *passwdfile*`.dir`. The `mkpasswd` command will exit with a non-zero exit code if any errors are detected. Otherwise it will print the number of entries processed and the length of the longest entry.

The presence of the hashed files is optional. They are not required for correct operation of `getpwnam()` or `getpwuid()` but will improve their performance on systems with moderate to large sized `/etc/passwd` files.

Diagnostics

The data base is already up to date.
If the specified file has been modified less recently than the hashed files.

Database not created.
When used with the `-u` option and the database does not already exist.

Files

passwdfile`.dir`	Database filename
passwdfile`.pag`	Database filename

See Also

getpwent(3), passwd(5), vipw(8)

mkproto (8)

Name

mkproto – construct a prototype file system

Syntax

/etc/mkproto special proto

Description

The `mkproto` command is used to bootstrap a new file system. First a new file system is created using `newfs`(8). The `mkproto` command is then used to copy files from the old file system into the new file system according to the directions found in the prototype file *proto*. The prototype file contains tokens separated by spaces or new lines. The first tokens comprise the specification for the root directory. File specifications consist of tokens giving the mode, the user-id, the group id, and the initial contents of the file. The syntax of the contents field depends on the mode.

The mode token for a file is a 6 character string. The first character specifies the type of the file. (The characters **–bcd** specify regular, block special, character special and directory files respectively.) The second character of the type is either **u** or **–** to specify set-user-id mode or not. The third is **g** or **–** for the set-group-id mode. The rest of the mode is a three digit octal number giving the owner, group, and other read, write, execute permissions. See `chmod`(1).

Two decimal number tokens come after the mode; they specify the user and group ID's of the owner of the file.

If the file is a regular file, the next token is a pathname whence the contents and size are copied.

If the file is a block or character special file, two decimal number tokens follow which give the major and minor device numbers.

If the file is a directory, `mkproto` makes the entries **.** and **..** and then reads a list of names and (recursively) file specifications for the entries in the directory. The scan is terminated with the token **$**.

A sample prototype specification follows:

```
d--777 3 1
usr    d--777 3 1
       sh     ---755 3 1 /bin/sh
       ken    d--755 6 1
              $
       b0     b--644 3 1 0 0
       c0     c--644 3 1 0 0
       $
$
```

Restrictions

You can only run `mkproto` on virgin file systems. It should be possible to copy files into existent file systems.

See Also

dir(5), fs(5), fsck(8), newfs(8)

mop_mom(8)

Name

mop_mom – MOP down-line/up-line load listener

Syntax

/etc/mop_mom

Description

The `mop_mom` command listens for down-line load and up-line dump requests on behalf of your local ULTRIX node. The `mop_mom` command, `/etc/mop_mom`, is usually included in the `/etc/rc.local` file. When a down-line load or up-line dump request is received from a target node, `mop_mom` spawns the loader, `mop_dumpload`, to process the load request.

To process a load request, the loader requires the name of the load file. During a down-line load process, the target system can either request that the loader search the nodes data base for the name of the load file, or it can specify an image name as part of its request. If the target system specifies an image name, the loader does not search the data base for the file before processing the load request.

If no absolute pathname is associated with the image file name, the loader searches for the file using the primary default path, `/usr/lib/mop`. If the image file is not in `/usr/lib/mop`, it uses the secondary default, `/usr/lib/dnet`. The names of the files in `/usr/lib/dnet` must be in lowercase with a `.sys` extension. Otherwise, the loader interprets the file specification literally without appending the `.sys` extension. For example, if you give the loader the file name `/usr/lib/mop/LOADME`, it attempts to open a file with an exact match.

The loader must have the following information before it can accept and process a load request:

Ethernet Physical Address
If the name of the image is not in the program request message itself, the Ethernet physical address of the target node must be specified in the nodes data base entry for the requesting target node. For further information, see `addnode`(8). The same is true if the environment variable, LOADUMP_SECURE is enabled.

Image File Name
The loader must be given the name of the image file to down-line load. If the image file name is specified in a target node's down-line load request, then the loader uses that file. Otherwise, the file name must be defined in the requesting node's data base entry. For further information, see `addnode`(8).

Dump File Name
The loader must be given the name of the dump file that will contain the up-line dump image of the requesting node's memory. The dump file name must be defined in the requesting node's data base entry. For further information, see `addnode`(8).

Note that the System Manager can force the loader to search the nodes data base, even when the target node specifies a file name. The loader verifies that an entry exists for the requesting node before proceeding with the down-line load. Any file name specified in the request message supersedes a file name specified in the data base.

To force the nodes data base search, use the following syntax before executing `/etc/mop_mom`:

```
# setenv LOADUMP_SECURE on
```

To permanently enable this feature, enter the following line in the `/etc/rc.local` file after the section on local daemons:

```
LOADUMP_SECURE=on /etc/mop_mom
```

The `mop_mom` command is installed in `/etc`. The down-line loader, `mop_dumpload`, is installed in `/usr/lib/dnet`.

The file `/usr/lib/mop` is the primary default pathname for all image files and `usr/lib/dnet` is the secondary default pathname. The names of the image files in `/usr/lib/dnet` must be in lowercase with a `.sys` extension.

Examples

```
/etc/mop_mom
```

This command is usually included in the `/etc/rc.local` file, which causes `mop_mom` to listen for down-line and up-line load requests.

Restrictions

The `mop_mom` command does not turn on network devices. When executed, it listens only on those devices that were previously started by network utilities such as `ifconfig` or DECnet `ncp`. Generally, you can ensure that `mop_mom` performs properly by putting it at the end of your rc.local file.

Currently, `mop_mom` only supports RSX, a.out, and VMS downline load images.

Files

`/usr/lib/dnet`	Down-line loader
`/usr/lib/mop`	Primary default pathname for all image files
`/usr/lib/dnet`	Secondary default pathname

See Also

addnode(8), ccr(8), getnode(8), load(8), remnode(8), trigger(8)
Guide to Ethernet Communication Servers

Name

mount, umount – mount or unmount file systems

Syntax

/etc/mount [*options*] [*device*] [*directory*]

/etc/umount [*options*] [*device*] [*directory*]

Description

This is a general description of the `mount` command. Additional `mount` descriptions are provided to define the `mount` syntax and options for the NFS and UFS file systems.

Each invocation of the `mount` command announces to the system that a file system is present on the device *device*. The file system may be local or remote. File *directory* must exist as a directory file. It becomes the name of the newly mounted file system root.

If invoked without arguments, `mount` prints the list of mounted file systems.

Physically write-protected disks and magnetic tape file systems must be mounted read-only or an error will occur at mount time.

General users can only mount file systems with certain restrictions. For example, the user, other than the superuser, performing the mount must own the directory *directory*. Furthermore, no users other than the superuser can execute `setuid` or `setgid` programs on the mounted file systems. In addition, users other than the superuser cannot access block or special character devices such as `rra0g` on the mounted file systems.

The `umount` command announces to the system that the removable file system previously mounted on the specified directory is to be removed. Only the person who mounted a particular file system or the superuser can unmount the file system again.

Options

–a	Reads the file `/etc/fstab` and mounts, or unmounts, all file systems listed there.
–f	Fast unmount. The `-f` option has no meaning for local file systems and directories. However, for remote file system types (such as NFS), the `-f` option causes the client to unmount the remotely mounted file systems and directories without notifying the server. This can avoid the delay of waiting for acknowledgment from a server that is down.
–o *options*	Specifies a string that is passed to the kernel and used by the specific file system's mount routine in the kernel. For specific options, refer to the file system-specific `mount` description, such as `mount`(8nfs).
–r	Indicates that the file system is to be mounted read only. To share a disk, each host must mount the file system with the **–r** option.
–t *type*	Specifies the type of file system is being mounted. When used with the `-a` option, the `-t` option mounts all file systems of the given type

found in the `/etc/fstab` file. For specific file system types, refer to the file system-specific `mount` description, such as `mount`(8nfs).

–v	Tells what did or did not happen. (Verbose flag)

The options for `umount` are:

–a	Unmounts all mounted file systems. It may be necessary to execute `umount -a` twice to accomplish unmounting of all mounted file systems.
–v	Tells what did or did not happen. (Verbose flag)

Restrictions

Mounting corrupted file systems will crash the system.

Files

`/etc/fstab`	File systems information table

See Also

getmnt(2), mount(2), fstab(5), fsck(8), mount(8nfs), mount(8ufs)

Name

mount, umount – mount and unmount a Network File System (NFS)

Syntax

/etc/mount –t nfs [**–r –v –o** *options*] *device directory*

/etc/umount [**–v**] *directory*

Description

The `mount` command allows you to mount a file system or directory onto a directory. Once a file system or directory has been mounted, it is treated as a file system.

The argument *device* can have one of the following forms:

host:remote_name

remote_name@host

The *remote_name* is the name of a file system or subtree of a file system that has been exported by *host*. The file *directory* must exist and must be a directory. It becomes the name of the newly mounted file system.

General users can mount file systems with certain restrictions in addition to those listed in `mount`(8).

The `umount` command unmounts the remote file system that was previously mounted on the specified directory.

Options

–f Fast unmount. The `-f` option has no meaning for local file systems and directories. However, for remote file system types (such as NFS), the `-f` option causes the client to unmount the remotely mounted file systems and directories without notifying the server. This can avoid the delay of waiting for acknowledgment from a server that is down.

–r Indicates that the file system is to be mounted read only.

–v Tells what did or did not happen. (Verbose flag)

–o *options* Specifies *options* as a sequence of comma-separated words from the list below. The defaults are:

**rw,hard,intr,retry=10,000,timeo=11,retrans=4, \
port=NFS_PORT,pgthresh=64**

Defaults for *rsize* and *wsize* are set by the kernel. The NFS options are:

bg If the first mount attempt fails, retry the mount in the background the number of times specified (the default is 10,000 times).

hard Retry the NFS operation (not the `mount`) request until server responds. The **hard** option applies after the `mount` has succeeded.

intr	Allow hard mounted file system operations to be interrupted.
nintr	Disallow hard mounted file system operations to be interrupted.
noexec	Binaries cannot be executed from this file system.
nosuid	The `setuid` and `setgid` programs may not be executed from this file system.
pgthresh=##	Set the paging threshold for this file system in kilobytes.
port=*n*	Set server IP port number to *n*.
retrans=*n*	Set number of NFS operation retransmissions (not the `mount`) to *n*. The **retrans=** option applies after the `mount` has succeeded.
retry=*n*	Set number of mount failure retries to *n*. The **retry=** option applies to the `mount` command, itself.
ro	Read-only.
rsize=*n*	Set read buffer size to *n* bytes.
rw	Read/write.
soft	Return an error if the server does not respond to the NFS operation (not the `mount`) request. The **soft** option applies after the `mount` has succeeded.
timeo=*n*	Set NFS timeout to *n* tenths of a second.
wsize=*n*	Set write buffer size to *n* bytes.

The following options affect how quickly you see updates to a file or directory that has been modified by another host. Increasing these values will give you slightly better performance. Decreasing these values decreases the time it takes for you to see modifications made on another host. If you are the only modifier of files under this mount point, you may increase these values.

acdirmin=*n*	Hold cached directory attributes for at least *n* seconds. The default is 30 seconds.
acdirmax=*n*	Hold cached directory attributes for no more than *n* seconds. The default is 60 seconds. The maximum value allowed is 3600.
acregmin=*n*	Hold cached file attributes for at least *n* seconds. The default is 3 seconds.
acregmax=*n*	Hold cached file attributes for no more than *n* seconds. The default is 60 seconds. The maximum value allowed is 3600.
actimeo=*n*	Set all four attributes cache timeout values to *n*.
noac	Do not set attribute caching. This is equivalent to **actimeo=0**.

The bg option causes mount to run in the background if the server's mountd daemon does not respond. The mount command attempts each request **retry=***n* times before giving up. Once the file system is mounted, each NFS request made in the kernel waits **timeo=***n* tenths of a second for a response. If no response arrives, the time-out is multiplied by 2 and the request is retransmitted. When **retrans=***n* retransmissions have been sent with no reply, a soft mounted file system returns an error on the request and a hard mounted file system retries the request. If a hard mounted file system was mounted with the intr option, an operation within that file system that is retrying (for example, the server is down) can be interrupted. File systems that are mounted rw (read-write) should use the hard option. The number of bytes in a read or write request can be set with the rsize and wsize options.

The option for umount is:

–v Tells what did or did not happen. (Verbose flag)

Restrictions

If the directory on which a file system is to be mounted is a symbolic link, the file system is mounted on the directory to which the symbolic link refers, rather than being mounted on top of the symbolic link itself.

The /etc/nfs_mount and /etc/nfs_umount commands should only be invoked by the mount and umount commands. Users (and superusers) should not invoke the nfs_mount and nfs_umount commands.

Examples

The mount command invokes nfs_mount to do its work. A sample mount command is:

```
# mount -t nfs -o hard,pgthresh=100 server:/usr /usr
```

To mount the remote file system /usr/src onto the local directory /mnt with 1k transfer size, type:

```
# mount -t nfs -o soft,rsize=1024,wsize=1024 serv:/usr/src /mnt
```

To mount the remote directory /usr/src/code onto the local directory /usr/src, type:

```
# mount -t nfs serv:/usr/src/code /usr/src
```

To hard mount a remote file system called /usr/src onto the local directory /usr/src, type:

```
# mount -t nfs -o hard serv:/usr/src /usr/src
```

Files

/etc/fstab	File system information file
/etc/nfs_mount	NFS-specific mount program
/etc/nfs_umount	NFS-specific unmount program

See Also

getmnt(2), mount(2), fstab(5), exports(5nfs), mount(8), umount(8), mountd(8nfs)

mount(8ufs)

Name

mount – mount the local ULTRIX File System (UFS)

Syntax

/etc/mount [**–r –v –o** *options*] *device directory*

Description

The `mount`(8ufs) command announces to the system that a file system is present on the device *device*. The specified device must be a local device. The file *directory* must exist and it must be a directory. It becomes the name of the newly mounted file system.

If invoked with **–r**, the device *device* is mounted on *directory* read only.

To further protect from system crashes, only file systems that have been cleanly checked by `fsck`(8) are mounted. In emergency situations, the superuser can override this requirement by using the `force` option as shown below.

General users can mount file systems with certain restrictions in addition to those listed in `mount`(8). The file system must have the clean byte set. To ensure the clean byte is set, run the `fsck`(8) command on the file system first. You can also try the mount and if it fails, then run `fsck` and then try the mount again.

Note that the user must have execute permissions on the device.

A successful ufs-mount may generate the following warning message:

```
"Warning, device has exceeded xxx threshold, fsck(8) is advised"
```

where xxx is which metric was exceeded to cause the clean byte timeout factor to reach zero. See `fsck`(8) for an explanation of the timeout algorithm.

Options

–o *options* Specifies *options* as a sequence of comma-separated words from the list below.

force	The superuser can force the mounting of unclean file systems.
nodev	Block and character special devices cannot be accessed from this file system.
noexec	Binaries cannot be executed from this file system.
nosuid	The `setuid` and `setgid` programs may not be executed from this file system.
pgthresh=##	Set the paging threshold for this file system in kilobytes. The default is 64 kilobytes.
sync	All writes are immediately written to disk (synchronously) as well as to the buffer cache. For the `sync` option to be meaningful, the file system must be mounted with write permissions.

Physically write-protected disks and magnetic tape file systems must be mounted read only or an error will occur at mount time. You should use the `force` flag only in single-user mode when repairing or recovering damaged file systems.

Examples

The `/etc/mount` command calls `ufs_mount` to do its work and is the preferred interface. A sample `mount` command is:

```
# mount -t ufs -o nodev,nosuid,noexec,pgthresh=100 /dev/ra0g /usr
```

Restrictions

The `/etc/ufs_mount` command should only be invoked by the `mount` command. Users (and superusers) should not invoke the `ufs_mount` command.

Files

`/etc/ufs_mount`
UFS-specific mount program

See Also

getmnt(2), mount(2), fsck(8), mount(8)

mountd(8nfs)

Name

mountd – NFS mount request daemon

Syntax

/etc/mountd [**–i**] [**–d**] [**–s**]

Description

The `mountd` daemon must be run on NFS servers to process NFS mount protocol requests. It reads the `/etc/exports` file to determine which file systems and directories are available to which machines and users, and makes this information available to the operating system. The machine names listed in the `/etc/exports` file may optionally contain the local BIND domain name. For more information on BIND, see the *Guide to the BIND/Hesiod Service*. To see which clients have file systems or directories mounted, use the `showmount` command.

Options

–i Turns on verification of the Internet address of the client against the server's `hosts` database for mounts and unmounts. The default is no address verification.

–d If you are running the BIND/Hesiod service, after checking the Internet address, `mountd` will verify that the host requesting a mount or unmount is in the server's domain.

–s If you are running the BIND/Hesiod service, after checking the Internet address, `mountd` will verify that the host requesting a mount or unmount is in the server's domain or a subdomain.

See Also

exports(5nfs), mount(8nfs), nfsd(8nfs), showmount(8nfs)
Guide to the BIND/Hesiod Service

Name

mtx – generic magtape exerciser

Syntax

/usr/field/mtx [*options*] **–a***dev*
/usr/field/mtx [*options*] **–s***dev*
/usr/field/mtx [*options*] **–l***dev*
/usr/field/mtx [*options*] **–v***dev*

Description

The `mtx` exerciser will write, read, and validate random data on the specified magnetic tape device from beginning of tape (BOT) to end of tape (EOT). There are four record length modes in which to run the `mtx` exerciser. The modes are short (512 bytes), long (10240 bytes), variable (512-20480 bytes), and all of these three in sequence.

The exerciser will run until <CTRL/C> or a kill -15 *pid* is sent to the process.

A logfile is made in `/usr/field` for you to examine and then remove. If there are errors in the logfile, check the `/usr/adm/syserr/syserr.<hostname>` file, where the driver and kernel error messages are saved.

An enhanced tape exerciser called `tapex` provides more comprehensive tape testing than this exerciser. Refer to `tapex`(8) for a complete description.

Options

The *mtx* options are:

–h	Print help message for the `mtx` command.
–o*file*	Save diagnostic output in *file*.
–t*i*	Run time in minutes (*i*). The default is to run until the process receives a <CTRL/C> or kill -15 *pid*.
–r*j*	Record length for long-record test. May range from 1 - 20480; the default is 10240 bytes.
–f*k*	Size of file in records. The default is –1, go to end-of-tape.

Arguments

You must specify one of the following function flags and its argument to the `mtx` exerciser.

–a*dev*	Perform short, long, and variable-length record tests on the *dev*, a raw device name and unit number. For example, **–armt0h**.
–s*dev*	Perform short (512-byte) record length test. The *dev* argument is a raw device name and unit number. For example, **–srmt0h**.
–l*dev*	Perform long (10240-byte) record length test. The *dev* argument is a raw device name and unit number. For example **–lrmt0h**.

–v*dev*	Perform variable record length test (records vary from 512 bytes to 20480 bytes). The *dev* argument is a raw device name and unit number. For example, **–vrmt0h.**

Restrictions

If there is a need to run a system exerciser over an NFS link or on a diskless system there are some restrictions. For exercisers that need to write into a file system, such as `fsx`(8), the target file system must be writable by root. Also the directory, in which any of the exercisers are executed, must be writable by root because temporary files are written into the current directory. These latter restrictions are sometimes difficult to overcome because often NFS file systems are mounted in a way that prevents root from writing into them. Some of the restrictions may be overcome by copying the exerciser to another directory and then executing it.

The following restrictions apply to the SCSI tape drives. The Magnetic Tape Exerciser (MTX) runs the tape in start/stop mode; that is, the tape does not stream. Therefore, MTX should not run for extended periods of time (two hours maximum run time). In addition, MTX does not handle the end of tape properly.

Examples

This example runs short, long, and variable-length tests on rmt0h until the process receives a <CTRL/C> or kill -15 *pid*:

```
% /usr/field/mtx -armt0h
```

The following example runs a long-record length test on rmt0h for 240 minutes in the background:

```
% /usr/field/mtx -lrmt0h -t240 &
```

See Also

tapex(8)
Guide to System Exercisers

Name

named – Internet name domain server daemon

Syntax

/usr/etc/named [–d *level#*] [**–p** *port#*] [**–b** *bootfile*][**–n**][**–a** *type.version*]

Description

The `named` daemon is the Internet domain name server for the BIND/Hesiod service. Without any arguments, `named` reads the default boot file `/etc/named.boot` and any initial data from the BIND/Hesiod data base files. `Named` then listens for queries.

The boot file specifies where the BIND/Hesiod server is to get its initial data. See the Example section.

The master data files consist of entries of the following form:

`$include` *file*
$origin *domain*
domain ttl addr-class entry-type resource-record-data

The `include` entry is useful for separating data into separate files. The `origin` entry is useful for placing more than one domain in a data file. It can also be used to set the reverse network number origin. The fields are:

file	This is the name of the file to be included.
domain	This is the domain name. An at sign (@) signifies the current origin. A name refers to the standard domain name. If the domain name does not end with a period, the current origin is appended to the domain. A domain name ending with a period is the complete BIND domain name (fully qualified) and thus does not get an extension appended to it.
ttl	This field is an optional integer specifying the time to live. If no time to live is specified, the default is obtained from the SOA entry.
addr-class	This field is the object class type. There are three classes:

IN	Objects connected to the DARPA Internet
HS	Hesiod naming service data
ANY	All classes

entry-type The most common entries for this field are listed below. The `resource-record-data` field, however, must correspond with the entry type:

A	Host address
CNAME	Canonical name for an alias
HINFO	Host information
MX	Mail exchanger
NS	Authoritative name server

PTR	Domain name pointer
SOA	Start of a zone of authority
TXT	Hesiod text
WKS	Well-known service description

The following signals have the specified effect when sent to the server `named` process using the `kill` command:

SIGXFSZ	Causes the server to reload only the databases that have changed.
SIGHUP	Causes the server to read `named.boot` and reload database.
SIGINT	Dumps the current data base and cache to `/var/tmp/named_dump.db`.
SIGIOT	Dumps `named` statistics to `/var/tmp/named.stats`.
SIGUSR1	Turns on debugging. Each time the SIGUSR1 signal is issued, the debug level increments by one. Debugging information is dumped to `/var/tmp/named.run`.
SIGUSR2	Turns off debugging.

Options

–b *bootfile* Names of the boot file. If no boot file is specified, the default is `/etc/named.boot`.

–d *level#* Prints debugging information. A number after the **–d** option determines the level of messages printed. It is a good idea to run the `named` daemon with the –d option in the background.

–p *port#* Specifies the port number. The default is the standard port number listed in the `/etc/services` file.

–n Runs `named` in network safe mode. All HS class queries sent over the network are authenticated. Non-authenticated queries for HS information from hosts other than local host are not answered. Non-authenticated requests for zone transfers are ignored.

–a *type.version* Specifies the default authentication type. When authenticated queries are formed, `named` needs to know what type of authentication to use. The *type* parameter specifies the form of authentication. The *version* parameter specifies the version of the *type* to use. The supported forms of authentication are:

type	Kerberos
version	One

Examples

The following is an example of a boot file:

```
;
;       boot file for name server
;
; type     domain                    source file or host
;
```

```
primary    cities.us             hosts.db
;
primary    2.10.in-addr.arpa     hosts.rev
;
primary    0.0.127.in-addr.arpa  named.local
;
secondary  cc.cities.us          10.2.0.78 128.32.0.10
;
; load the cache data last
cache                            named.ca
```

Entries beginning with a semicolon are comment lines. In this example, the first line that is not a comment specifies that this system is the primary authoritative BIND server for the domain `cities.us`. This line also specifies that the file `hosts.db` contains authoritative data for the `cities.us` domain. Domain names in the file `hosts.db` are relative to the origin, such as `cities.us` in the preceding example.

The second and third non-comment entries (beginning with `primary`) show the `in-addr.arpa` domain in reverse order. This allows address to name mapping.

The fourth non-comment line specifies that all authoritative data under `cc.cities.us` is to be transferred from the primary master server at IP address `10.2.0.78` to the secondary server. If the transfer fails, the secondary server will then try the master server at address `128.32.0.10`. There can be up to 10 IP addresses listed.

The cache entry specifies that the data in `named.ca` is to be placed in the cache. This would include well known data such as the locations of root domain servers.

Files

`/var/dss/namedb/named.boot`	Name server configuration boot file
`/etc/named.pid`	Process ID number
`/var/tmp/named.run`	Debug output
`/var/tmp/named_dump.db`	Dump of the BIND server's cache

See Also

kill(1), signal(3c), resolver(3), hesiod.conf(5), resolv.conf(5)
Guide to the BIND/Hesiod Service

Name

named-xfer – pull BIND/Hesiod zones from another server

Syntax

/usr/etc/named-xfer –z *zone_to_transfer* **–f** *db_file* **–s** *serial_no* [**–d** *debug_level*] [**–l** *debug_log_file*] [**–t** *trace_file*] [**–p** *port*] [**–n**] [**–a** *auth_type.auth_ver*] *servers...*

Description

The `named` transfer daemon, `/usr/etc/named-xfer`, is a server that is usually run by the `named` daemon, `/usr/etc/named`, but it can also be run manually with the given arguments. The `named` transfer daemon runs on a BIND/Hesiod secondary server and pulls BIND/Hesiod zones from a primary server. This daemon is not run by default, nor can it be started up from `inetd`(8c).

Options

–z *zone_to_transfer*	This option is required to pull a zone. The *zone_to_transfer* argument specifies the name of the BIND/Hesiod zone that the `named-xfer` daemon will transfer, for example, dec.com.
–f *db_file*	This option is required to pull a zone. The *db_file* argument specifies the name of the file into which the pulled zone information is placed.
–s *serial_no*	This option is required to pull a zone. The *serial_no* argument should be set to the current serial number of the SOA record for the zone *zone_to_transfer*. If *serial_no* is set to 0, the zone is always pulled.
–d *debug_level*	The *debug_level* argument sets the debug level and determines the amount of debug information to be displayed.
–l *debug_log_file*	The *debug_log_file* argument specifies the file that will contain any debug messages from the zone pull.
–t *trace_file*	The *trace_file* argument specifies the file that will contain a trace from the zone pull.
–p *port*	The *port* argument specifies the port that will be used instead of the default nameserver port listed in `/etc/services`.
–n	This option must be used when the `named` daemon is running in the network safe mode. It indicates that the zone pull must be authenticated.
–a *auth_type.auth_ver*	This option must be used if the `named` daemon is running in the network safe mode. The *auth_type* argument indicates what type of authentication to use and the *auth_ver* argument indicates what version of the authentication type to use. Currently, the *auth_type* must be "kerberos" and the auth_ver must be "one".

servers...	The servers argument is a list of Internet addresses from which to pull a zone. If the first host cannot be ddreached, the `named` transfer daemon will try to pull the zone from the next host listed.

See Also

services(5), named(8), kerberos(8krb)
Guide to the BIND/Hesiod Service
Guide to Kerberos

ncheck (8)

Name

ncheck – generate names from i-numbers

Syntax

/etc/ncheck [**–i** numbers] [**–a**] [**–s**] [filesystem]

Description

For most normal file system maintenance, the function of `ncheck` is subsumed by `fsck`(8).

The `ncheck` command with no argument generates a pathname vs. i-number list of all files on a set of default file systems. Names of directory files are followed by '/.'. The **–i** option reduces the report to only those files whose i-numbers follow. The **–a** option allows printing of the names '.' and '..', which are ordinarily suppressed. The **–s** option reduces the report to special files and files with set-user-ID mode; it is intended to discover concealed violations of security policy.

A file system may be specified.

The report is in no useful order, and probably should be sorted.

Diagnostics

When the filesystem structure is improper, '??' denotes the 'parent' of a parentless file and a pathname beginning with '...' denotes a loop.

See Also

sort(1), dcheck(8), fsck(8), icheck(8)

Name

netsetup – network set up program

Syntax

/etc/netsetup [install]

Description

The `netsetup` command allows you to add your system to a local area network (LAN). It also enables you to update the `/etc/hosts` and `/etc/hosts.equiv` files.

To add your system to a LAN, bring the system to single-user mode. Then run the `netsetup` command with the `install` option and answer the questions. The network manager should be able to provide you with the information that you need. You will need to know your network number and your host number. In addition, you need to find out your Internet Protocol (IP) broadcast address and if the LAN uses all zeros (0) or ones (1). Furthermore, you have to know if the LAN uses subnet routing. If the answer is yes, then you need to know how many bits of the host part of the Internet address are used to specify the subnet part of the network address.

NOTE

If your LAN uses subnet routing, remember to enter the subnet number as part of the host number.

After the `netsetup` command has completed, reboot the system.

To add or modify hosts in the `/etc/hosts` or `/etc/hosts.equiv` files, run the `netsetup` command without the `install` option and answer all the questions. There is no need to bring the system to single-user mode to update these files.

Files

```
/etc/hosts
/etc/hosts.equiv
/etc/networks
/etc/rc.local
```

See Also

Guide to Local Area Networks

Name

netx – TCP/IP net exerciser

Syntax

/usr/field/netx [**–h**] [**–t***n* B] [**–p***m*] *nodename*

Description

The `netx` exerciser sets up a stream socket connection to the `miscd`(8c) server in the TCP/IP internet domain. With connection made, the exerciser writes ramdom data to the `miscd` server; the server loops the data back to `netx`, and the data is read and verified against the data written out.

The `netx` exerciser runs in conjunction with the `miscd`(8c) server.

The `netx` exerciser by default will use the port number of the echo service in the `/etc/services` file. Make sure that the TCP echo service is enabled in the `/etc/inetd.conf` file (no '#' in front of the service). The exerciser will run until <CTRL/C> or kill -15 *pid* is sent to the process. The *nodename* is the remote or local system host name running the `miscd`(8c) server.

Options

The `netx` options are:

–h	Print the help message for the `netx` command.
–t*n*	Run time in minutes (*n*). The default is to run until the process receives <CTRL/C> or kill -15 *pid.*
–p*m*	Specify port number to use in internet domain (*m* < 32768). Note that this option is not used with the `miscd`(8c) server, so you should never have to use this option.

Restrictions

If there is a need to run a system exerciser over an NFS link or on a diskless system, there are some restrictions. For exercisers that need to write into a file system, such as `fsx`(8), the target file system must be writable by root. Also the directory, in which any of the exercisers are executed, must be writable by root because temporary files are written into the current directory. These latter restrictions are sometimes difficult to overcome because often NFS file systems are mounted in a way that prevents root from writing into them. Some of the restrictions may be overcome by copying the exerciser to another directory and then executing it.

Examples

The following example exercises the TCP/IP network from the local host to node **keel** until a <CTRL/C> or kill -15 *pid* is received:

```
% /usr/field/netx keel
```

The following example exercises the TCP/IP network from the local host to node **photon** for 180 minutes in the background:

```
% /usr/field/netx -t180 photon &
```

See Also

Guide to System Exercisers

newfs(8)

Name

newfs – construct a new file system

Syntax

/etc/newfs [–v] [–n] [mkfs-options] special disk-type

Description

The `newfs` command is a front-end to the `mkfs`(8) program. The `newfs` program looks up the type of disk a file system is being created on in the disk description file `/etc/disktab`, calculates the appropriate parameters to use in calling `mkfs`, then builds the file system by forking `mkfs` and, if the file system is a root partition and installs the necessary bootstrap program in the initial 16 sectors of the device.

Options

The **–n** option prevents the bootstrap program from being installed.

If the **–v** option is supplied, `newfs` will print out its actions, including the parameters passed to `mkfs`.

Options which may be used to override default parameters passed to `mkfs` are:

–s size The size of the file system in sectors.

–b block-size
The block size of the file system in bytes.

–f frag-size
The fragment size of the file system in bytes.

–t #tracks/cylinder

–c #cylinders/group
The number of cylinders per cylinder group in a file system. The default value used is 16.

–m free space %
The percentage of space reserved from normal users; the minimum free space threshold. The default value used is 10%.

–r revolutions/minute
The speed of the disk in revolutions per minute (normally 3600).

–S sector-size
The size of a sector in bytes (almost never anything but 512).

–i number of bytes per inode
This specifies the density of inodes in the file system. The default is to create an inode for each 2048 bytes of data space. If fewer inodes are desired, a larger number should be used; to create more inodes a smaller number should be given.

Files

`/etc/disktab`	For disk geometry and file system partition information
`/etc/mkfs`	To actually build the file system
`/usr/mdec/bootblks`	For boot strapping program

See Also

disktab(5), fs(5), diskpart(8), fsck(8), format(8v), mkfs(8), tunefs(8)
''A Fast File System for UNIX,'' *ULTRIX Supplementary Documents, Volume 3: System Manager*

newfs (8)

Name

newfs – construct a new file system

Syntax

/etc/newfs [**–v**] [**–n**] [**–N**] [**mkfs-options**] **special disk-type**

Description

The `newfs` command is a front-end to the `mkfs`(8) program. The **-N** option is used to run in no update mode. In this mode, `mkfs` will not write to `special.` The `newfs` program looks up the type of disk a file system is being created on in the disk description file `/etc/disktab,` calculates the appropriate parameters to use in calling `mkfs,` then builds the file system by forking `mkfs` and, if the file system is a root partition and installs the necessary bootstrap program in the initial 16 sectors of the device.

If there is no disk description for the specified disk type in the `/etc/disktab` file, the `newfs` program will use the `creatdiskbyname(3x)` subroutine to derive disk geometry information from the controlling device driver. This functionality is provided for MSCP and SCSI disks.

Options

–n Prevents the bootstrap program from being installed.

–v Instructs `newfs` to print out its actions, including the parameters passed to `mkfs.`

Options which may be used to override default parameters passed to `mkfs` are:

–s size The size of the file system in sectors.

–b block-size
The block size of the file system in bytes.

–f frag-size
The fragment size of the file system in bytes.

–t #tracks/cylinder
–c #cylinders/group
The number of cylinders per cylinder group in a file system. The default value used is 16.

–m free space %
The percentage of space reserved from normal users; the minimum free space threshold. The default value used is 10%.

–r revolutions/minute
The speed of the disk in revolutions per minute (normally 3600).

–S sector-size
The size of a sector in bytes (almost never anything but 512).

–i number of bytes per inode
This specifies the density of inodes in the file system. The default is to create an inode for each 2048 bytes of data space. If fewer inodes are

desired, a larger number should be used; to create more inodes a smaller number should be given.

Files

`/etc/disktab`	For disk geometry and file system partition information
`/etc/mkfs`	To actually build the file system
`/usr/mdec/vaxboot`	For boot strapping program

See Also

disktab(5), fs(5), chpt(8), fsck(8), format(8v), creatediskbyname(3x), mkfs(8), tunefs(8)
''A Fast File System for UNIX,'' *ULTRIX Supplementary Documents, Volume 3: System Manager*

nfsd (8nfs)

Name

nfsd – NFS server daemon

Syntax

/etc/nfsd [nservers]

Description

The `nfsd` daemon starts the specified number of NFS server daemons which handle file system requests from clients. The *nservers* argument tells `nfsd` how many file system server daemons to start. This number should be based on the load expected on this server. The `nfsd` daemon is normally invoked at boot time through the `/etc/rc.local` file.

Examples

```
/etc/nfsd 4     /* Start four daemons on a large machine /*
```

See Also

exports(5nfs), mountd(8nfs)

Name

nfsportmon - turn on or off the port monitor

Syntax

/etc/nfsportmon *option*

Description

The `nfsportmon` command turns the port monitor on or off at the kernel level. By default, the port monitor is turned off.

When the port monitor is on, NFS server port checking is activated. This checking increases system security by ensuring that file access requests were generated by the client kernel rather than by a forged application program, for example.

Only the superuser or root can run `nfsportmon`.

Options

on	Turn on the port monitor.
off	Turn off the port monitor.

Files

`/etc/rc.local`

See Also

Guide to the Network File System

nfssetlock (8nfs)

Name

nfssetlock – turn on or off the NFS locking service

Syntax

/etc/nfssetlock *option*

Description

The `nfssetlock` command turns the NFS locking service on or off at the kernel level. By default, the NFS locking service is turned off.

If the NFS locking service is turned off, local locking is active. Local locking only coordinates the dispersal of advisory locks to local files and file regions. The NFS locking service coordinates the dispersal of advisory locks to both local and remote files and file regions through the `fcntl` and `lockf` primitives.

Only the superuser or root can run `nfssetlock.` The best way to run `nfssetlock` is through the `nfssetup` command, which provides an interactive means for enabling and disabling the NFS locking service.

Restrictions

If you run `nfssetlock` manually, you must issue it while the system is in single-user mode. Otherwise, the locking information could become lost during the transition between local and NFS locking.

Options

on	Turn on the NFS locking service.
off	Turn off the NFS locking service.

Files

`/etc/rc.local`

See Also

fcntl(2), lockf(3), nfssetup(8nfs)
Guide to the Network File System

Name

nfssetup – set up the network file system (NFS)

Syntax

/etc/nfssetup

Description

The `nfssetup` command is an interactive facility that allows you to set up or modify NFS on your system. A local area network must be set up on your system before you can set up NFS.

The `nfssetup` command allows you to either enable or disable the NFS locking service. In addition, the `nfssetup` command appends entries to the `/etc/exports` and `/etc/fstab` files.

NOTE

> To remove entries from the `/etc/exports` or `/etc/fstab` you must edit them by hand. The `nfssetup` command only appends entries to these files.

You can run `nfssetup` while the system is in multiuser mode. To run `nfssetup`, type the following and then answer the questions:

```
# /etc/nfssetup
```

When `nfssetup` has completed, reboot the system.

If you use `dms` to set up your system as a diskless server before running `nfssetup`, `dms` will automatically call the `nfssetup` utility to set up a default NFS server environment.

Files

```
/etc/exports
/etc/fstab
/etc/rc.local
```

See Also

dms(8), biod(8nfs), mountd(8nfs), nfsd(8nfs), rwalld(8c)
Guide to the Network File System

nfsstat(8nfs)

Name

nfsstat – display Network File System (NFS) statistics

Syntax

/usr/etc/nfsstat [**–cnrsz**] [**vmunix.***n*] [**core.***n*]

Description

The `nfsstat` command displays statistical information about the Network File System (NFS) and Remote Procedure Call (RPC) interfaces in the kernel. It can also be used to reinitialize this information. If you do not specify any options, `nfsstat` displays the information as though all the options were specified, but `-z`.

The statistics are reinitialized to zero each time the system reboots.

Options

–c Display the client information. The client side NFS and RPC information is displayed. You can combine this option with the `-n` and `-r` options to print client NFS or client RPC information only.

–s Display the server information. The server side NFS and RPC information is displayed.

–n Display the NFS information. The NFS information for both the client and server side is displayed. You can combine this option with the `-c` and `-s` options to print client or server NFS information only.

–r Display the RPC information. The RPC information for both the client and server side is displayed. You can combine the `-r` option with the `-c` and `-s` options to print client or server RPC information only.

–z Reinitialize the statistics to zero. You can combine this option with any of the above options to reset particular sets of statistics to zero after printing them. You must have write permission on `/dev/kmem` to use this option.

core.*n* This is the core image, which is usually stored in the directory `/usr/adm/crash`. If no `core` is specified, the default is `/dev/mem`.

vmunix.*n* This is the kernel image, which is usually stored in the directory `/usr/adm/crash`. If no `vmunix` is specified, the default is `/vmunix`.

Here is a sample of `nfsstat` output with no options specified:

```
# nfsstat

Server rpc:
calls      badcalls   nullrecv   badlen     xdrcall
1312142    0          0          0          0

Server nfs:
calls      badcalls
1312142    0
null       getattr    setattr    root       lookup     readlink   read
0  0%      319612 24% 1220   0%  0  0%      795544 60% 5857   0%  163962 12%
```

```
wrcache     write       create      remove      rename      link        symlink
0  0%       7294  0%    165  0%     239  0%     75  0%      74  0%      0  0%
mkdir       rmdir       readdir     fsstat
0  0%       0  0%       17612  1%   334  0%


Client rpc:
calls       badcalls    retrans     badxid      timeout     wait        newcred
30156       40          256         0           296         0           0

Client nfs:
calls       badcalls    nclget      nclsleep
30143       40          30156       0
null        getattr     setattr     root        lookup      readlink    read
0  0%       5833 19%    21  0%      0  0%       17630 58%   420  1%     3455 11%
wrcache     write       create      remove      rename      link        symlink
0  0%       475  1%     84  0%      10  0%      4  0%       0  0%       0  0%
mkdir       rmdir       readdir     fsstat
2  0%       0  0%       1423  4%    786  2%
```

Of the client RPC statistics, each field is as follows:

calls	The total number of client RPC calls successfully begun.
badcalls	The total number of unsuccessful (badly formed) RPC calls.
retrans	The number of times that RPC calls were transmitted.
badxid	The number of times a reply transaction ID did not match the request transaction ID.
timeout	The number of times a request was made but not answered.
wait	The number of times the client system had to sleep because the client structure was busy.
newcred	This field is never used, and is therefore always 0.

Of the client NFS statistics, each field is as follows:

calls	The total number of client NFS calls successfully begun.
badcalls	The total number of unsuccessful (badly formed) NFS calls.
nclget	The number of times a client structure was successfully acquired. The client structure is where clients keep track of an outstanding RPC call.
nclsleep	The number of times all client structures were busy. Since there are six client structures, `nclsleep` is the number of times that there were six operations in progress when a seventh one arrived and had to wait until one of the client structures was freed.

Of the server RPC statistics, each field is as follows:

calls	The total number of RPC calls received by NFS daemons.
badcalls	The number of badly formed RPC calls.
nullrecv	The number of empty RPC calls.
badlen	The number of RPC calls with too small of a body.
xdrcall	The number of RPC calls that failed to decode in XDR.

Of the server NFS statistics, each field is as follows:

calls	The total number of NFS calls dispatched by an NFS daemon.
badcalls	The number of badly formed NFS requests.

The remaining fields provide counts of the completed NFS operations. Here are their descriptions:

null	This is the number of null operations. If the software is working properly, this field should be zero.
getattr	This is the number of file attributes that were retrieved. In the example above, there were 319,612, or 24% on the server.
setattr	This is the number of file attributes that were stored.
root	This field is not used and should always be zero.
lookup	This is the number of times that a directory pathname was looked up.
readlink	This is the number of times a symbolic link was read.
read	This is the number of times data was read from a file.
wrcache	This field is not used and should always be zero.
write	This is the number of times data was written to a file.
create	This is the number of times a new file was created.
remove	This is the number of times a file was removed.
rename	This is the number of times a file was renamed.
link	This is the number of times a hard link was created.
symlink	This is the number of times a symbolic link was created.
mkdir	This is the number of times a directory was created.
rmdir	This is the number of times a directory was removed.
readdir	This is the number of times a directory was read.
fsstat	This is the number of times that file system attributes and statistics were retrieved.

Files

`/vmunix`	System namelist
`/dev/kmem`	kernel memory

Name

nrglbd – non-replicating global location broker (GBL) daemon

Syntax

/etc/ncs/nrglbd [**–version**]

Description

The global location broker (GLB), enables clients to locate servers on a network or internet. The GLB database stores the locations (that is, the network addresses and port numbers) where server processes are running. The GLB maintains this database and provides access to it.

The `/etc/ncs/nrglbd` daemon should run as a background process. It requires no options or arguments. A Local Location Broker daemon (`llbd`) must be running on the local host when `nrglbd` is started.

You can run only one `nrglbd` on a network or internet.

On ULTRIX systems, `nrglbd` is typically started by a line in `/etc/rc` such as the following:

```
/etc/ncs/nrglbd& echo -n ' nrglbd' > /dev/console
```

Options

–version	Display the version of the Network Computing Kernel (NCK) that this `nrglbd` belongs to but do not start the daemon. (NCK is part of the Network Computing System (NCS) on which DECrpc is based.)

Restrictions

This section discusses the procedure to follow if the system running the `nrglbd` is taken off-line.

If you restart `nrglbd` on the same system and no server on any other system changed state, all things should run as before. If, however, an application tries to contact a server that is no longer running or which has different port numbers, the application will fail. The application also will not see any new server registrations.

If a copy of `/etc/ncs/glbdbase.dat` is not available, you must create an up to date version of the file before restarting `nrglbd.` To do so, use `lb_admin` to query the `llbd` for registration data on every system running an DECrpcserver and then use `lb_admin` to register all DECrpc servers with the GLB on the new host. Then restart `nrglbd.`

See Also

lb_admin(1ncs), llbd(8ncs)
Guide to the Location Broker

Name

ntalkd – remote user communication server

Syntax

/etc/talkd

Description

The `ntalkd` command is the server that notifies a user that somebody else wants to initiate a conversation. It acts as a repository of invitations, responding to requests by clients wishing to meet to hold a conversation. In normal operation, a client (the caller) initiates a rendezvous by sending a CTL_MSG to the server of type LOOK_UP. (For further information, see `protocols/talkd.h.)` This causes the server to search its invitation tables to check if an invitation currently exists for the caller to speak to the client specified in the message. If the lookup fails, the caller then sends an ANNOUNCE message causing the server to broadcast an announcement on the client's login ports requesting contact. When the client responds, the local server uses the recorded invitation to respond with the appropriate meeting address and the programs of the caller and client he called establish a stream connection through which the conversation takes place.

Restrictions

This daemon is used in conjunction with the current version of `talk`(1). The `talkd`(8c) command is used for systems running Version 2.2 or earlier of `talk`(1).

See Also

talk(1), write(1)

Name

ntpd – network time protocol (NTP) daemon

Syntax

/usr/etc/ntpd [**–a** *threshold*][**–c** *file*][**–d**][**–D** *level*][**–l**][**–n**][**–s**]

Description

The University of Maryland's `ntpd` daemon synchronizes the local clock with a set of distributed time servers. The `ntpd` daemon distributes accurate, reliable time from the best time source available at your site to hosts on wide area networks (WAN) and local area networks (LAN). The three recommended time sources in decreasing order of accuracy are: Internet NTP service, local radio clock, and wristwatch. Note that the `ntpd` daemon does not require time servers to be on the same LAN as time clients, and does not create a heavy broadcast load on the network.

NOTE

If the NTP servers are not on your Local Area Network (LAN), you must run the `routed` daemon before running the `ntpd` daemon. To run `routed`, remove the number signs (#) from in front of the following lines in your `/etc/rc.local` file:

```
#[ -f /etc/routed ] && {
#       /etc/routed & echo 'routed'              >/dev/console
#}
```

The `routed` daemon will be invoked when you reboot your system. To start `routed` without rebooting, type the following on the command line:

```
/etc/routed
```

For information on setting up the network time services, see the *Guide to System and Network Setup*.

The `ntpd` daemon automatically splits the nodes running the `ntpd` daemon into a dynamically reconfigurable hierarchy of nodes. The nodes at the top level of the hierarchy (low stratum numbers) are connected to the most accurate sources available. This information is transferred to the lower-level nodes (higher stratum numbers) which set their clocks based on the calculated offset from a remote server, and then distribute this time to lower levels of the hierarchy.

The `ntpd` daemon provides a solution for distributing time to a large number of individual workstations. It can also be used in conjunction with a master `timed` daemon to distribute NTP time to workstations running `timed`. If `timed` is run with the `-E` and `-M` options on at least one system that is also running `ntpd`, then all other systems on the network running `timed` can receive time updates from a host running `ntpd`. Although `timed` is easier to set up on clients, NTP is recommended because it is more accurate and more secure.

Normally, the `ntpd` daemon is invoked at boot time from the `/etc/rc.local` file. When `ntpd` is started, it reads configuration information from the `/etc/ntp.conf` file, unless you have specified another configuration file with the

–c option. The configuration file either specifies the list of NTP servers with which this host should synchronize, or identifies this host as a local reference clock. See the `ntp.conf`(5) reference page for more information on the `/etc/ntp.conf` configuration file.

The `ntpd` daemon uses the `adjtime`(2) system call to gradually adjust the local clock for small clock offsets (< 0.128 seconds). If the local clock lags the time on the server by more than 0.128 seconds, the `settimeofday`(2) system call is used to make a forward step adjustment of the local clock. Clocks are never stepped backwards; they are adjusted gradually, which can take a very long time. Therefore, it is important to initialize the time using the `ntp` command before running the `ntpd` daemon.

Options

–a *threshold*
Sets the threshold (in seconds) which limits how far the `ntpd` daemon can change the local clock. By default, the threshold is 1000 seconds. This is set to avoid propagating major mistakes throughout the network. If you specify the string `any` instead of a number, the `ntpd` daemon can change the local clock by any amount.

–c *file*
Specifies a configuration file for the `ntpd` daemon. By default, the configuration file is `/etc/ntp.conf`.

–d Increments the debug level by one. The `-d` option can be specified more than once. Higher debug levels provide more diagnostic information.

–D *level*
Sets the debug level to the specified value.

–l Causes the `ntpd` daemon to log a message each time the local clock is adjusted. Specify this option only if you want to gather statistical information to analyze the local clock behavior. If the `-l` option is set, a message may be logged every two minutes. Messages are logged to `/usr/spool/mqueue/syslog`.

–n Inhibits the `ntpd` program from being swapped out of memory. Using the `-n` option is recommended for both time servers and time clients.

–s Prevents the `ntpd` daemon from altering the time on the local host. The `ntpd` daemon participates as an NTP server with the `-s` flag set, but it does not change the time of the local host.

Examples

Before starting the `ntpd` daemon, either manually or from the `/etc/rc.local` file, you must edit the `/etc/ntp.conf` file with the appropriate information. If your system is a client you must specify the time servers for it to query. If it is a time server, you must specify the time servers with which it peers. See the `ntp.conf`(5) reference page for more information.

To start the `ntpd` daemon manually (on a time client), enter the following commands:

```
# /etc/rdate -s
# /usr/etc/ntp -s -f server1 server2 server3
# /usr/etc/ntpd -n
```

The /etc/rdate command initializes your time to the average network time. The /usr/etc/ntp command further refines the initial time to the NTP time. The servers specified on the command line are the same as those specified in the /etc/ntp.conf file.

To start the ntpd daemon from the /etc/rc.local file (on a time client), place the following entries in the /etc/rc.local file. Multiple servers are included in case one of the servers crashes, or is brought down. The servers specified in the /etc/rc.local file are the same as those specified in the /etc/ntp.conf file. These entries should be placed after the syslog entry:

```
[-f /etc/syslog] && {
     /etc/syslog   & echo -n ' syslog'          >/dev/console
}
[-f /etc/rdate] && {
     /etc/rdate -s   & echo -n ' rdate'         >/dev/console
}
[-f /usr/etc/ntp] && {
     /usr/etc/ntp -s -f server1 server2 server3 \
                   & echo -n ' ntp'             >/dev/console
}
[-f /usr/etc/ntpd] && {
     /usr/etc/ntpd -n & echo -n ' ntpd'         >/dev/console

}
```

Diagnostics

The ntpd daemon logs errors, major state changes, and statistics reports using the syslog daemon; the log entries appear in the file /usr/spool/mqueue/syslog with the word ntpd: on each relevant line. Normal log entries show when ntpd gains or loses synchronization with a lower-stratum host. Also, once an hour ntpd issues a ntpd: stats: entry that gives information about its state.

Once an hour, if ntpd is synchronized, it updates the /etc/ntp.drift file. This file shows the estimated clock drift for each of the past 5 hours, with the most recent hour listed first. (The 6th number in this file is the number of hours ntpd has been running). To convert the drift values to parts per million (ppm), divide them by 4096 and multiply by 1000000. For example, +0.0107001161 means that ntpd estimates that the clock is drifting by about 2.61 ppm, or is losing about 0.226 seconds per day. [(2.61/1000000) * 24 * 60 * 60 = 0.226]. Negative drift values mean that the clock is gaining time. If ntpd is working, your computer's clock should be accurate to within a few seconds per day.

Another diagnostic tool is the ntpdc command. You can use this to look at any host running ntpd. The following command line returns the state of the remote host's ntpd server:

```
% /usr/etc/ntpdc hostname
```

The value that is returned for the offset should contain values not greater than 100 milliseconds. See the ntpdc(8) reference page for more information.

If the `ntpd` daemon sets the time as frequently as every 10 minutes (indicated by messages in the `/usr/spool/mqueue/syslog` file), then you should kill the `ntpd` daemon, remove the `/etc/ntp.drift` file, run the `ntp` command to initialize the time, and restart the `ntpd` daemon.

If your system clock is ahead of the server time by more than 1 second, you should kill the `ntpd` daemon, remove the `/etc/ntp.drift` file, run the `ntp` command to initialize the time, and restart the `ntpd` daemon.

If your clock is more than 1000 seconds off from the server time, and you did not specify the `-a any` option, `ntpd` will not change your system time. Rather, `ntpd` will repeatedly log messages to the `/usr/spool/mqueue/syslog` file, indicating that the time is too far off to reset.

See Also

ntp(1), adjtime(2), settimeofday(2), ntp.conf(5), ntpdc(8), timed(8)
RFC 1129—Internet Time Synchronization: the Network Time Protocol
Guide to System and Network Setup
Introduction to Networking and Distributed System Services

Name

ntpdc – monitor operation of the NTP daemon, ntpd

Syntax

/usr/etc/ntpdc [**–n**][**–v**] *host1* | *IPaddress1* ...

Description

The `ntpdc` command sends a query to the `ntpd` daemon running on each of the hosts listed on the command line. The `ntpd` daemon on each responding host sends information about the current calculated offset between its time and the time of each of its NTP servers or peers. The `ntpdc` command formats the information on the standard output.

NOTE

You can specify hosts by either host name or Internet address. The hosts that you specify must either exist in the `/etc/hosts` file, or in the master `hosts` database, if the database is being served to your system by BIND/Hesiod or Yellow Pages.

The `ntpdc` command by default generates a terse, table-style report. If you specify the `-v` option, the `ntpdc` command generates a verbose report.

Options

–n Prints Internet addresses, instead of host names, of the servers or peers. By default, the Internet addresses of the responding hosts and the names of their servers or peers are printed.

–v Prints a verbose report for each of the servers or peers of the responding host.

Examples

Terse Report:

The following is a typical terse report generated in response to the command:

```
% /usr/etc/ntpdc 555.5.55.55
```

The host 555.5.55.55 is an NTP client, with the servers `server1`, `server2`, and `server3` specified in its `/etc/ntp.conf` file. The information returned is about `server1`, `server2`, and `server3`.

Address		**Strat**	**Poll**	**Reach**	**Delay**	**Offset**	**Disp**
(rem)	**(lcl)**						
.*server1*	555.5.55.55	1	64	377	53.0	-65.0	5.0
**server2*	555.5.55.55	1	256	377	155.0	-4.0	16.0
+*server3*	555.5.55.55	2	64	377	16.0	-61.0	3.0

The fields are interpreted as follows:

– , + , . or *

A minus sign (–), plus sign (+), or dot (.) indicates a pre-configured peer (see the `ntp.conf`(5) reference page). The asterisk (*) indicates which pre-configured peer (if any) is currently being used for synchronization.

(rem)	The remote host name or Internet address of a peer or server of the responding host.
(lcl)	The Internet address of the responding host that was specified on the `ntpdc` command line.
Strat	The current operating stratum level of the peer or server. Since the NTP hierarchy can change dynamically the stratum levels may change. Lower stratum levels correspond to higher accuracy.
Poll	Current polling interval in seconds for this peer or server. Polling intervals change dynamically.
Reach	Reachability in response to the last 8 polls (value of 8-bit shift register with bits entering from the end furthest to the right).
Delay	The estimated round-trip delay in milliseconds for NTP message exchanges between the responding host and this peer or server. Delay is calculated from the previous 8 polls.
Offset	The estimated offset between the peer or server's time and the responding host's time in milliseconds. This value is calculated from the previous 8 polls.
Disp	The current estimated value of dispersion in milliseconds for this peer's offset/delay pair. Dispersion is used by the `ntpd` daemon in the clock selection algorithm. Increasing values of dispersion are associated with decreasing quality of the estimate.

Verbose Report:

When the `-v` option is given, a verbose report for each of the servers or peers of each of the hosts specified on the command line is generated.

The following is a typical verbose report generated in response to the following command line:

```
% /usr/etc/ntpdc -v 111.11.111.11

Neighbor address 555.55.5.55 port:123  local address 111.11.1.11
Reach: 0377 stratum: 2, precision: -7
dispersion: 2.000000, flags: 1301, leap: 0
Reference clock ID: [22.22.2.22] timestamp: 7e5aa1a9.2add5d0b
hpoll: 10, ppoll: 10, timer: 1024, sent: 85 received: 90
Delay(ms)    20.00  20.00  28.00  29.00  20.00  39.00  29.00  28.00
Offset(ms)    5.00   6.00   5.00  -1.00  -2.00   0.00   3.00   5.00

        delay: 20.000000 offset: 5.000000 dsp 2.000000
-------------------------------------------------------------------
```

The fields are interpreted as follows:

Neighbor address
The address and port number of one NTP server, followed by the Internet address of the responding host (`local address`).

Reach: Reachability in response to the last 8 polls (value of 8-bit shift register with bits entering from the end furthest to the right).

stratum: The current operating stratum level of the peer or server. Since the NTP hierarchy can change dynamically the stratum levels may change. Lower stratum levels correspond to higher accuracy.

precision:
The precision of this clock, given in seconds as a power of 2. If precision is equal to –7, that means that the precision is 2**–7, or 1/128 seconds. The `ntpd` daemon automatically determines the precision of each clock based on the kernel variable `HZ`.

disp: The current estimated value of dispersion in milliseconds for this peer's offset/delay pair. Dispersion is used by the `ntpd` daemon in the clock selection algorithm. Increasing values of dispersion are associated with decreasing quality of the estimate.

flags: *nn* The `flags` parameter is used by the `ntpd` daemon clock selection process.

leap: *flag* The leap second indicator. Non-zero if there is to be a leap second inserted in the NTP timescale. The bits are set before 23:59 on the day of insertion and reset after 00:00 on the following day.

Reference clock ID: *address*
If the NTP server is synchronized by a radio/satellite timecode receiver, this field is `WWV`, `WWVB`, or `GOES`. If the NTP server is the local reference clock, this field is `LOCL`. Finally, this field can be the `[internet_address]` of the most accurate NTP server currently serving the responding host.

timestamp: *nn*
The local time, in hex-timestamp format, when the local clock of the server was last updated.

hpoll: *n* The host poll interval which is the maximum interval between messages transmitted to the server, in seconds as a power of 2. For example, a value of 6 indicates an interval of 64 seconds.

ppoll: *n* The peer poll interval which is the maximum interval between messages sent by the server, in seconds as a power of 2. For example, a value of 6 indicates an interval of 64 seconds.

timer: *nn* The current poll rate in seconds.

sent: *nn* The number of NTP packets sent to this server by the responding host.

received: *nn*
The number of NTP packets received from this server by the responding host.

Delay and Offset
The round-trip delay and estimated clock offset for the last eight NTP packet exchanges. If there are fewer than eight valid samples, the delay field will be zero.

delay: *est-delay* **offset:** *est-offset* **dsp:** *n*
Estimated delay, offset, and dispersion calculated from the above 8 samples. See RFC 1129 for how to calculate the estimated delay, offset, and dispersion.

Diagnostics

hostl: connection refused
Check whether the `ntpd` daemon is running on `host1`.

host2: unknown
The `ntpdc` command cannot resolve the host name `host2`. Check that `host2` exists in the `/etc/hosts` file, or that it exists in the master `hosts` database, if the database is being served by BIND/Hesiod or Yellow Pages.

If a server is listed in the host's `/etc/ntp.conf` file, but does not appear in the `ntpdc` report, it is possible that the `ntpd` daemon on the responding host can not resolve the server names in the `/etc/ntp.conf` file. Check that the server exists in the responding host's `/etc/hosts` file or in the master `hosts` database, if the database is being served to the responding host by BIND/Hesiod or Yellow Pages.

See Also

ntp.conf(5), ntp(8), ntpd(8)
RFC 1129—Internet Time Synchronization: the Network Time Protocol
Introduction to Networking and Distributed System Services

Name

opser – interactive program for system maintenance

Syntax

opser

Description

The `opser` program is a utility that simplifies normal system and incremental backup. It is invoked by default when the system operator logs in to the operator account. The system then runs the `/opr/opser` utility in place of a shell. The `opser` utility prompts the operator to bring the system to single-user mode for file maintenance. This includes a file system consistency check and file system backup, and then prompts to restart multiuser mode.

Network `opser` allows the operator to perform `opser` commands on a slave system. The master system name must be in the `/.rhosts` file on the slave system and the slave name in the in the master system's `/.rhosts` file. If necessary, the operator can escape to the shell, execute commands, and return. Online help is provided. The `opser` program allows remote file system backup, file system check, restart and both local and remote shell command execution.

Restrictions

- The local `opser` utility supports only tape as a backup media.
- The remote `opser` utility allows backup to files, disks, or tapes.
- The `opser` utility must be run from the system console; if run from a terminal, its operation is limited.
- The `opser` utility does not contain file restore capability.
- The operator must escape to the shell to perform this function.

Diagnostics

Included in the utility as it runs.

Files

```
/opr/opser
/opr/backup
/opr/dobackup
```

See Also

Guide to System Backup and Restore

pac(8)

Name

pac – printer/plotter accounting information

Syntax

/etc/pac [**–Pprinter**] [**–pprice**] [**–s**] [**–r**] [**–c**] [*name* ...]

Description

The `pac` command reads the printer/plotter accounting files, accumulating the number of pages (the usual case) or feet (for raster devices) of paper consumed by each user, and printing out how much each user consumed in pages or feet and dollars. If any *names* are specified, statistics are only printed for those users; usually, statistics are printed for every user who has used any paper.

Options

–P	Causes accounting to be done for the named printer. Normally, accounting is done for the default printer (site dependent) or the value of the environment variable **PRINTER** is used.
–p	Causes the value *price* to be used for the cost in dollars instead of the default value of 0.02.
–c	Causes the output to be sorted by cost; usually the output is sorted alphabetically by name.
–r	Reverses the sorting order.
–s	Causes the accounting information to be summarized on the summary accounting file; this summarization is necessary since on a busy system, the accounting file can grow by several lines per day.

Restrictions

The relationship between the computed price and the actual price is unknown.

Files

`/usr/adm/?acct`	Raw accounting files
`/usr/adm/?_sum`	Summary accounting files

Name

pfconfig – configure packet filter parameters

Syntax

/usr/etc/pfconfig [**+/–p[romisc]**] [**–b[acklog]** *nnn*] [**–a[ll]**] *[interface-name ...]*

Description

The `pfconfig` command allows the system manager to configure certain parameters of the packet filter driver (see `packetfilter`(4)). These parameters are configured separately for each interface; the interfaces are specified by name on the command line (for example, `ln0`, and `ni1`). If more than one interface is specified, they are all given the same settings. Alternatively, you can specify to configure all the packet-filter interfaces on the system.

You can set the following parameters with `pfconfig`:

+promisc Allows packet filter users to set the interface into promiscuous mode (receives all packets). Whenever there is at least one packet filter descriptor open with the ENPROMISC mode bit set, the interface will be in promiscuous mode. When no such descriptors are in use, the interface will be returned to normal mode.

–promisc The interface will no longer be put into promiscuous mode on behalf of packet filter users; if the interface is in promiscuous mode when this command is given, it is returned to normal mode. (The superuser may use `ifconfig`(8c) to control promiscuous mode, overiding the mode set by non-superusers. This is the default setting.)

–backlog *nnn* Sets the maximum backlog (packet filter input queue length) for non-superuser descriptors to the specified number. When a descriptor is opened, it is given a queue length limit of two. An application can increase this backlog using the EIOCSETW ioctl request. Superusers are allowed to increase their backlog up to a system-wide maximum; non-superusers are allowed to increase their backlog only up to the maximum set by this program. Note that allowing too large a backlog may result in vast amounts of kernel memory being tied up in the packet filter driver queues.

If no configuration parameters are specified, the `pfconfig` command displays the current packet filter configuration for the network interface(s).

Only the superuser may use this command to change the configuration.

Examples

On a system used for network monitoring, one might put this line into `/etc/rc.local`:

```
/usr/etc/pfconfig -a +promisc -backlog 64
```

This allows users to run promiscuous network monitoring applications, with a maximum input queue length per application of 64 packets, on any interface in the system.

Diagnostics

Messages indicating the specified interface do not exist; an attempt to set a maximum backlog less than 1 or greater than the system-wide maximum; the user tried to alter an interface's configuration but is not privileged.

See Also

netstat(1), intro(4n), packetfilter(4), ifconfig(8c), rc(8)

Name

pfstat – print packet filter status information

Syntax

pfstat [**–cdfkpqsv01234567**] [*system* [*corefile*]]

Description

The `pfstat` command interprets the data structures of the packet filter driver `packetfilter(4)`. If *system* is given, the required namelist is taken from there; otherwise, it is taken from `/vmunix`. If *corefile* is given, the data structures are sought there, otherwise in `/dev/kmem`. (If *corefile* is a core dump, the `-k` option must be given.)

Options

If no options are given, then all are assumed (except for the verbose option, `-v`).

c	Counts. Displays various counts (per ethernet unit) including number of packets sent and received, the number of packets dropped due to full input queues, the number of packets not wanted by any filter, and the number of packets missed by the interface.
d	Descriptors. Displays OpenDescriptors for each minor device.
f	Filters. Displays packet filters for each minor device.
k	Specifies the corefile is a crash dump, not a running system's `/dev/kmem`.
p	Parameters. Displays device parameters including device type, header and address lengths, maximum transmission units (MTU), and interface and broadcast. addresses.
q	QueueElements. Displays the QueueElements.
s	Scavenger. Displays the FreeQueue and Scavenger statistics.
v	Verbose. Displays information for minor devices not actually in use and complete queue information, only if this flag is given.
<digit>	Limits output to information about specified units. If no digits are given, all units are displayed.

Output Format

This section describes the information displayed in the output of the `pfstat` command under the headings `AllDescriptors`, `Filters`, and `QueueElts`.

`AllDescriptors`

#	Minor device number for open descriptor.
LOC	Descriptor location.
LINK-QUEUE	Forward link to other descriptors.

STATE	Blank, or one of: **wait** waiting for input, indefinite wait **timed** waiting for input, timed wait **tout** has timed out
WAIT-QUEUE	Addresses of ''Queue Elements'' for waiting packets.
NQ'D	Number of packets queued for input/maximum for this queue.
TOUT	Timeout duration in clock ticks (if the `-v` [Verbose] option is not given, then times may be expressed as minutes [with a trailing ''m''], hours [with a trailing ''h''], or simply ''long'', to keep the columns lined up.)
MODE	Shows which mode bits are set for the minor device; each bit is encoded as a single character: **H** ENHOLDSIG **B** ENBATCH **T** ENTSTAMP **P** ENPROMISC **N** ENNONEXCL **?** An unknown mode bit is set.
SIG	Signal number to be delivered when a packet arrives.
PROC	Process to be signaled when a packet arrives.
PID	Process id which enabled the signal.

`Filters`

LOC	Location of descriptor.
DROPS	Count of ''recent'' drops for this filter.
PRI	Priority of filter.
LEN	Length of filter (in shortwords).
FILTER	See `packetfilter`(4) for interpretation of packet filters.

`QueueElts`

LOC	Location of queue element.
LINK-QUEUE	Forward and backward links.
COUNT	Packet size.
REF	Reference count for queue element.

FLAGS	Per-packet flag bits set; each bit is encoded as a single character:
	P ENSF_PROMISC
	B ENSF_BROADCAST
	M ENSF_MULTICAST
	T ENSF_TRAILER
	? An unknown flag bit is set.
DROP	Count of packets dropped between the time previous packet was queued and the time this packet was queued.
TIME	Approximate time this packet was received.

Restrictions

Some of the output is a bit cramped to fit on an 80-character line. It should be possible to get a less verbose but more readable listing.

Since things happen quickly, it is not likely that `pfstat` will provide a consistent view of a running system. It is mostly useful for analyzing static or slowly-varying problems, not transient ones.

Files

`/vmunix`	Namelist
`/dev/kmem`	Default source of tables

See Also

netstat(1), packetfilter(4), pfconfig(8c), pstat(8)

ping(8)

Name

ping – send ICMP ECHO_REQUEST packets to network hosts

Syntax

/etc/ping [*options*] *host* [*datasize* [*npackets*]]

Description

The DARPA Internet is a large and complex network of hardware connected together by gateways. The `ping` command utilizes the ICMP protocol's mandatory ECHO_REQUEST datagram to elicit an ICMP ECHO_RESPONSE from a host or gateway. ECHO_REQUEST datagrams (pings) have an IP and ICMP header, followed by a **struct timeval**, and then an arbitrary number of pad bytes used to fill out the packet. The length of the default datagram 64 bytes, but this may be changed using the command-line option.

Typing ''ping *host*'' without any options will either report ''*host* is alive'' or ''no answer from *host*''. To get more statistics use the **–l** option or one of the other options.

When using `ping` for fault isolation, it should first be run on the local host to verify that the local network interface is up and running. Then, hosts and gateways further and further away should be pinged. The `ping` command with options sends one datagram per second and prints one line of output for every ECHO_RESPONSE returned. No output is produced if there is no response. If an optional *npackets* is given, only that number of requests is sent. Round-trip times and packet loss statistics are computed. When all responses have been received or the program times out with *npackets* specified, or if the program is terminated with a SIGINT, a brief summary is displayed.

Options

-d Turns on SO_DEBUG flag on the socket.

-l Gives more statistics than if `ping` is used without options. Long output.

-r Bypasses the normal routing tables and sends directly to a host on an attached network. If the host is not on a directly-attached network, an error is returned. This option can be used to ping a local host through an interface that has no route through it. For example, after the interface was dropped by `routed`(8c).

-v Lists ICMP packets other than ECHO RESPONSE that are received. Verbose output.

Restrictions

This program is intended for use in network testing, measurement, and management. It should be used primarily for manual fault isolation. Because of the load it could impose on the network, it is unwise to use `ping` during normal operations or from automated scripts.

See Also

netstat(1), ifconfig(8c)

portmap(8nfs)

Name

portmap – DARPA internet port to RPC program number mapper

Syntax

/usr/etc/portmap

Description

The `portmap` command is a server that converts RPC program numbers into DARPA protocol port numbers. When an RPC server is started, it will tell `portmap` what TCP or UDP port number it is listening on, and what RPC program numbers it is prepared to serve.

It is important to understand that `portmap` must first be invoked before attempting to run any RPC service. It must be running in order to use any RPC-based applications, including the NFS mount daemon, `mountd`(8nfs). When a client wishes to make an RPC call to a given program number, the RPC program will first contact `portmap` on the server machine to determine the port number where the RPC packets should be sent.

If `portmap` crashes, all servers must be restarted.

See Also

rpcinfo(8nfs)

Name

post – deliver a message

Syntax

/usr/new/lib/mh/post [–alias *aliasfile*] **[–filter** *filterfile*] **[–nofilter] [–format] [–noformat] [–msgid] [–nomsgid] [–verbose] [–noverbose] [–watch] [–nowatch] [–width** *columns] file [–help]*

Description

The `post` program is called by `send`(1mh) to deliver the message in `file` to local and remote users. In fact, all of the functions attributed to `send` on its manual page are performed by `post`, with `send` acting as a preprocessor. Thus, it is `post` which parses the various header fields, appends From: and Date: lines, and interacts with the `sendmail` transport system.

Normally, `post` would not be called directly by the user.

It searches the `To:`, `cc:`, `Bcc:`, `Fcc:`, and `Resent-xxx:` header lines of the specified message for destination addresses, checks these addresses for validity, and formats them so as to conform to ARPAnet Internet Message Format protocol, unless the `-noformat` flag is set. This will normally cause `@local-site` to be appended to each local destination address, as well as any local return addresses. The `-width columns` switch can be used to indicate the preferred length of the header components that contain addresses.

If a `Bcc:` field is encountered, its addresses will be used for delivery. The `Bcc:` field is removed from the message sent to original recipients. The copied recipients will receive an entirely new message with a minimal set of headers. Included in the body of the message will be a copy of the message sent to the original recipients. If `-filter filterfile` is specified, then this copy is filtered (re–formatted) prior to being sent to the blind recipients.

The `-alias` *aliasfile* switch can be used to specify a file that `post` should take aliases from. More than one file can be specified, each being preceded with `-alias`. In any event, the primary alias file is read first.

The `-msgid` switch indicates that a `Message-ID:` or `Resent-Message-ID:` field should be added to the header.

The `-verbose` switch indicates that the user should be informed of each step of the posting/filing process.

The `-watch` switch indicates that the user would like to watch the transport system's handling of the message (for example, local and `fast` delivery).

This command consults the envariable `$SIGNATURE` to determine the sender's personal name in constructing the `From:` line of the message.

The default settings for `post` are:

–alias /usr/new/lib/mh/MailAliases
```
-format
-nomsgid
-noverbose
```

```
-nowatch
-width 72
-nofilter
```

Files

`/usr/new/lib/mh/mtstailor`
: Tailor file

`/usr/new/mh/refile`
: Program to process Fcc:s

`/usr/new/lib/mh/mhl`
: Program to process Bcc:s

`/usr/new/lib/mh/MailAliases`
: Primary alias file

Profile Components

The `post` command does not consult the user's `.mh_profile`.

See Also

mhmail(1mh), send(1mh), mh–mail(5mh) and mh–alias(5mh)
Standard for the Format of ARPA Internet Text Messages (RFC 822)

Name

pstat – print system facts

Syntax

/etc/pstat –aixptufTk [*suboptions*] [*system*] [*corefile*]

Description

The `pstat` command interprets the contents of certain system tables. When specifying a *namelist* and *corefile* with the `pstat` command, it is necessary to use the `-k` option.

Options

-a Under `-p`, describe all process slots rather than just active ones.

-f Print the open file table with these headings:

LOC	The core location of this table entry.
TYPE	The type of object the file table entry points to.
FLG	Miscellaneous state variables encoded thus:

R	Open for reading
W	Open for writing
A	Open for appending
S	Shared lock
X	Exclusive use
I	Asychronous I/O notification
B	Block if inuse bit is set (shared line semaphore)

CNT	Number of processes that know this open file.
GNO	The location of the gnode table entry for this file.
OFFS/SOCK	The file offset, see `lseek`(2), or the core address of the associated socket structure.

-i Print the gnode table with the these headings:

LOC	The core location of this table entry.
FLAGS	Miscellaneous state variables encoded thus:

L	Locked
U	Update time, `fs`(5), must be corrected
A	Access time must be corrected
M	File system is mounted here
W	Wanted by another process (L flag is on)

T	Contains a text file
C	Changed time must be corrected
S	Shared lock applied
E	Exclusive lock applied
Z	Someone waiting for an exclusive lock
I	Inuse flag is set (shared line semaphore)

CNT	Number of open file table entries for this gnode.
DEV	Major and minor device number of file system in which this gnode resides.
RDC	Reference count of shared locks on the gnode.
WRC	Reference count of exclusive locks on the gnode. (This count can be > 1 if, for example, a file descriptor is inherited across a fork.)
GNO	I-number within the device.
MODE	Mode bits, see `chmod`(2).
NLK	Number of links to this gnode.
UID	User ID of owner.
SIZ/DEV	Number of bytes in an ordinary file, or major and minor device of special file.

-k Prevents the process created from becoming too large, thereby causing performance problems. If **-k** is given, then either *system* or *corefile* is indicated. If *corefile* is given, the tables are sought there. Otherwise they are sought in `/dev/kmem`. The required namelist is taken from `/vmunix` unless *system* is specified.

-p Prints the process table for active processes with these headings:

LOC	The core location of this table entry.
S	Run state encoded thus:

0	No process
1	Waiting for some event
3	Runnable
4	Being created
5	Being terminated
6	Stopped under trace

F	Miscellaneous state variables, or-ed together (hexadecimal):

00000001	Process is resident in memory
00000002	System process: swapper, pager, idle (RISC only), trusted path daemon
00000004	Process is being swapped out

00000008	Process requested swapout for page table growth
00000010	Traced
00000020	Used in tracing
00000040	Locked in by `plock`(2)
00000080	Waiting for page-in to complete
00000100	Protected from swapout while tranferring resources to another process
00000200	Used by `sigpause`(2)
00000400	Exiting
00000800	Protected from swapout while doing physical I/O
00001000	Process resulted from a `vfork`(2) which is not yet complete
00002000	Parent has received resources returned by `vfork`(2) child
00004000	Process has no virtual memory, as it is a parent in the context of `vfork`(2)
00008000	Process is demand paging data pages from its text gnode
00010000	Process has advised of sequential memory access
00020000	Process has advised of random memory access
00080000	Process has indicated intent to execute data or stack (RISC only)
00100000	POSIX environment: no SIGCLD generated when children stop
00200000	Process is owed a profiling tick
00400000	Used by `select`(2)
00800000	A login process
04000000	System V file lock applied
08000000	Fix-up of unaligned accesses is attempted (RISC only)
10000000	Process has done an `execve`(2)
20000000	The idle process (RISC only)

POIP	Number of pages currently being pushed out from this process.
PRI	Scheduling priority, see `setpriority`(2).
SIGNAL	Signals received (signals 1-32 coded in bits 0-31).
UID	Real user ID.
SLP	Amount of time process has been blocked.
TIM	Time resident in seconds; times over 127 coded as 127.

CPU	Weighted integral of CPU time, for scheduler.
NI	Nice level, see `setpriority`(2).
PGRP	Process number of root of process group (the opener of the controlling terminal).
PID	The process ID number.
PPID	The process ID of parent process.
ADDR	If in memory, the page frame number of the page table entries of the 'u-area' of the process. If swapped out, the position in the swap area measured in multiples of 512 bytes.
RSS	Resident set size – the number of physical page frames allocated to this process.
SRSS	RSS at last swap (0 if never swapped).
SIZE	Virtual size of process image (data+stack) in multiples of 512 bytes.
WCHAN	Wait channel number of a waiting process.
LINK	Link pointer in list of runnable processes.
TEXTP	If text is pure, pointer to location of text table entry.
CLKT	Countdown for real interval timer, `setitimer`(2) measured in clock ticks (10 milliseconds).
TTYP	Address of controlling terminal.
DMAP	Address of data segment dmap structure.
SMAP	Address of stack segment dmap structure.

-s Print the following information about the (1k byte) pages used for swap space:

- The number of pages reserved, but not necessarily allocated, by the system for currently executing processes.
- The number of pages used (physically allocated), including the number used for text images.
- The number of pages free, wasted, or missing. Free pages are pages that have not been allocated. Missing pages are usually allocated to argdev. Wasted pages indicate the amount of space lost because the swap space is fragmented.
- The number of pages available, which indicate the amount of space available for swapping.

-t Print table for terminals with these headings:

RAW	Number of characters in raw input queue.
CAN	Number of characters in canonicalized input queue.
OUT	Number of characters in output queue.
MODE	See tty(4).
ADDR	Physical device address.

DEL	Number of delimiters (newlines) in canonicalized input queue.
COL	Calculated column position of terminal.
STATE	Miscellaneous state variables encoded thus:

T	Line is timed out
W	Waiting for open to complete
O	Open
C	Carrier is on
B	Busy doing output
A	Process is awaiting output
X	Open for exclusive use
H	Hangup on close
S	Output is stopped (ttstop)
I	Inuse flag is set (shared line semaphore)
D	Open nodelay
G	Ignore carrier
N	Non-blocking I/O
Z	Asychronous I/O notification
L	Terminal line is in the process of closing
Q	Output suspended for flow control

PGRP	Process group for which this is controlling terminal.
DISC	Line discipline; blank is old tty OTTYDISC, ntty for NTTYDISC, or termio for TERMIODISC.

-T Prints the number of used and free slots in the several system tables and is useful for checking to see how full the system tables have become if the system is under a heavy load.

-u Print information about a user process. The next argument is its address as given by `ps`(1). The process must be in main memory, or the file used can be a core image and the address 0.

-x Print the text table with these headings:

LOC	The core location of this table entry.
FLAGS	Miscellaneous state variables encoded thus:

T	The `ptrace`(2) command in effect.
W	Text not yet written on swap device.
L	Loading in progress.
K	Locked.
w	Wanted (L flag is on).
F	Text structure on Freelist

P	Resulted from demand-page-from-gnode exec format. For further information, see `execve`(2).
l	Locked from being paged or swapped. For further information, see `plock`(2).
B	All attached processes being killed due to server write of a.out file.

DADDR	Address of text dmap structure in core.
CADDR	Head of a linked list of loaded processes using this text segment.
SIZE	Size of text segment, measured in multiples of 512 bytes.
IPTR	Core location of corresponding gnode.
CNT	Number of processes using this text segment.
CCNT	Number of processes in core using this text segment.
LCNT	Number of process locking this text segment.
POIP	Number of pages currently being pushed out in this text segment.
CMAP	The address of the last CMAP entry freed.

Restrictions

Information on processes can change while `pstat` is running. The picture it gives is a snapshot taken at a given time.

Files

`/vmunix`	Namelist
`/dev/kmem`	Default source of tables
`/dev/mem`	Used for the `-u` option

See Also

ps(1), stat(2), fs(5)

Name

quot – summarize file system ownership

Syntax

/etc/quot [*option* ...] [*filesystem*]

Description

The `quot` command prints the number of blocks in the named *filesystem* currently owned by each user. If no *filesystem* is named, a default name is assumed.

Options

–n	Causes the pipeline **ncheck filesystem | sort +0n | quot –n filesystem** to produce a list of all files and their owners.
–c	Prints three columns giving the file size in blocks, the number of files of that size, and the cumulative total of blocks in that size or smaller file.
–f	Prints the count of number of files as well as the space owned by each user.

Files

`/etc/passwd`	Used to get user names

The default file system varies with the system.

See Also

du(1), ls(1)

quotacheck(8)

Name

quotacheck – file system quota consistency checker

Syntax

/etc/quotacheck [**–v**] [**–f**] *filesystem* ...
/etc/quotacheck [**–v**] **–a**

Description

The `quotacheck` command examines each file system, builds a table of current disc usage, and compares this table against that stored in the disc quota file for the file system. If `quotacheck` detects any inconsistencies, it updates both the quota file and the current system copies of the incorrect quotas. Inconsistencies occur only if an active file system is checked.

Normally, `quotacheck` returns a warning if it cannot find a valid quota file. If you use the `-f` option, `quotacheck` creates a quota file automatically and then performs its normal functions.

If you use the `-a` flag in place of any file system name, `quotacheck` checks all the file systems listed in `/etc/fstab` to be read-write with disc quotas.

Normally `quotacheck` reports only modified quotas. If the `-v` option is supplied, `quotacheck` will indicate the calculated disc quotas for each user on a particular file system.

The `quotacheck` command expects each file system to be checked to have a quota file named `quotas` in the root directory. If none is present, `quotacheck` will ignore the file system.

The `quotacheck` command is normally run at boot time from the `/etc/rc.local` file, before enabling disc quotas with `quotaon`(8). For further information, see `rc`(8).

The `quotacheck` command accesses the raw device in calculating the actual disc usage for each user. Thus, the file systems checked should be quiescent while `quotacheck` is running.

Files

`/etc/fstab`	Default file systems

See Also

quota(2), setquota(2), quotaon(8)
"Disk Quotas in a UNIX Environment," *ULTRIX Supplementary Documents, Volume 3: System Manager*

Name

quotaon, quotaoff – turn file system quotas on and off

Syntax

/etc/quotaon [*option* ...] [*filesystem*]

/etc/quotaoff [*option* ...] [*filesystem*]

Description

The `quotaon` command informs the system that disc quotas should be enabled on one or more file systems. Specified file systems must have entries in `/etc/fstab` and be mounted. Quota files must be present in the root directory of the specified file systems and be named *quotas*. The quotas file is created with `quotacheck`(8). The quota file is updated by `edquota`(8).

Options

The option `-v` causes `quotaon` to print a message for each file system whose quotas are enabled. The `-a` argument may be used in stead of a list of file systems. It causes `quotaon` to enable quotas on all file systems in `/etc/fstab` marked read-write. This option is normally used at boot time to enable quotas.

The `quotaoff` command informs the system that disc quotas should be disabled on one or more file systems. The `-v` option forces a verbose message for each file system affected. The `-a` option forces all file systems in `/etc/fstab` to have their quotas disabled.

Files

`/etc/fstab`	File system table

See Also

setquota(2), fstab(5), edquota(8)
''Disk Quotas in a UNIX Environment,'' *ULTRIX Supplementary Documents, Volume 3: System Manager*

Name

radisk – Digital Storage Architecture (DSA) disk maintenance

Syntax

/etc/radisk

radisk –c *LBN length special*
radisk –e *special*
radisk –n *special*
radisk –r *LBN special*
radisk –s *LBN length special*

Description

The Digital Storage Architecture (DSA) disk maintenance program `radisk` provides the essential functions to manage DSA disk devices. The `radisk` command must be used on unmounted disk partitions to insure correct results.

The *LBN* is a decimal number that represents the logical block number as reported in the errorlog file. The *LBN* is the actual disk block number starting from the beginning of the disk.

The *length* is a decimal number that indicates how many (512 byte) blocks to process. The length specified may be **–1** to indicate the last block of the specified partition.

The *special* file specified is used with **–c, –e, –n,** and **–r** options and indicates an unmounted **c** partition of a character device special file.

Options

The following options may be set with `radisk`.

–c Clears a forced error indicator on the range of specified LBNs. The forced error condition indicates that the data in the disk block is bad. The disk block is good, but the data can not be read without getting an error detection code (EDC) error. This option will cause the forced error condition to be removed. After the forced error indicator is cleared, the EDC error will not be reported nor will the data be marked as bad.

All indications that the data is corrupt are lost. The data should be restored either by manual methods or with the `restore` command. The `radisk` command affects the integrity of the data on a disk and should be followed by a file system restore if data is affected.

–e Sets the exclusive access attribute associated with the specified disk. This attribute is provided by multihost controllers to restrict access to a disk to one host. The `radisk` command will return a failure status if the disk is already exclusively associated with another host or the underlying controller does not provide multihost support. If the command is issued to a disk that is currently mounted and the command fails, the disk will no longer be online to this host. For this reason the `-e` option should not be issued to a disk that is mounted.

–n Clears the exclusive access attribute associated with the specified disk. If the controller provides multihost support and the exclusive access attribute is not set for a particular disk, it would be possible for the disk to be accessed by more than one host. The `radisk` command will return a failure status if the disk is not currently and exclusively associated with this host or the underlying controller does not provide multihost support.

–r Replaces a block on the disk specified by LBN. See Restrictions.

–s Starts a scan for bad blocks on the specified area on a disk. Bad blocks are disk blocks that have data transfer errors to the extent that they cannot be relied on. When a bad block is found, it is replaced and the bad block's LBN is reported. The LBN specified with the `-s` option can be **0** to indicate the first block in the specified partition. If **0** is specified, however, the program starts searching from the first block of the partition. The `-s` option will accept any valid partition on the disk. This allows any partition to be scanned without scanning the entire disk and ensures that the specified partition is free of bad blocks. As an example, `/dev/rra3h` indicates the **h** partition of the third logical disk unit.

Diagnostics

The `radisk` command generates messages when the user is not privileged, when the LBN is not in the specified partition, and when the length exceeds the size of the partition.

Restrictions

You must be in single-user mode when using the `-c`, `-r`, and `-s` options of the `radisk` program. If you are in multiuser mode, `radisk` hangs the system and cannot be killed. If this happens, you must reboot.

The `-r` option is supported only with those DSA disks which use host-initiated dynamic bad block replacement.

The `-e` and `-n` options are only supported on controllers that provide multi-host support. These options are only supported on HSC Version 5.00 or later.

See Also

dkio(4), ra(4), sdc(4), chpt(8), mount(8), restore(8)
Guide to System Disk Maintenance

rarpd(8c)

Name

rarpd – reverse address resolution protocol (RARP) daemon

Syntax

/usr/etc/rarpd [*interface*] [**–v**] [**–n**] [**–f** *filename*]

Description

The `rarpd` daemon maps the Ethernet address of a machine to the machine's Internet Protocol (IP) address.

When `rarpd` is invoked, it reads the `/etc/ethers` file (by default) and waits to process a RARP request. The `/etc/ethers` file is checked every ten minutes for any changes. If the file has been modified, `rarpd` reads it again. You can disable this feature with the `-n` option. You can force a scan of the `/etc/ethers` file by sending the `rarpd` daemon a SIGHUP signal. See `signal`(3) for more information on SIGHUP.

The format of the `/etc/ethers` file is described in `ethers`(5). You can specify a file other than `/etc/ethers` with the `-f` option. The *interface* is the network interface on which the `rarpd` daemon should listen. The command `netstat -i` shows the correct interface or interfaces for your system. The `rarpd` daemon uses the first interface it finds, if you do not specify an interface. See the `netstat(1)` reference page for more information.

Because the `rarpd` daemon has been implemented with the Ethernet Packet Filter (see `packetfilter`(4)), you must configure your kernel with the packet filter option in order for `rarpd` to function properly. The packet filter detects RARP broadcast packets and passes them to `rarpd` for processing. The filter priority for rarpd is set to 28.

All messages from the `rarpd` daemon are directed to `syslog`.

Options

interface	Specifies the system's network interface.
–v	Causes `rarpd` to operate in verbose mode. This option logs details of RARP to `syslog`. The instance of a RARP request and its response are also logged.
–n	Disables checking of the `ethers`(5) file. By default, `rarpd` checks the `ethers` file once every ten minutes, and, if the file was modified, `rarpd` rereads the file. If you specify the `-n` option `rarpd` scans the `/etc/ethers` file once at startup time.
–f *filename*	Reads an alternate Ethernet address file.

Restrictions

The `rarpd` ignores all ARP requests sent encapsulated within a RARP packet. You should make all ARP request using the `arp` command.

The machine for which an IP address is being requested must be present in the server's `/etc/hosts` file.

Diagnostics

The following message is logged to `syslog` if the Packet Filter is not configured in your kernel:

```
Packet Filter is not configured in /vmunix
```

The following messages are printed to your screen if the Packet Filter is not configured in your kernel:

```
rarpd: cannot find symbol Pfilt_read in /vmunix
option PACKETFILTER does not appear to be configured in
       your kernel.
```

Files

`/etc/ethers`	Database that maps Ethernet addresses to hostnames

See Also

packetfilter(4), ethers(5), hosts(5), arp(8c), ifconfig(8c), syslog(8c)
The Packet Filter: An Efficient Mechanism for User-Level Network Code

Name

rc – command script for auto-reboot and daemons

Syntax

/etc/rc
/etc/rc.local

Description

The `rc` command script controls the automatic reboot and `rc.local` is the script holding commands which are pertinent only to a specific site.

When an automatic reboot is in progress, `rc` is invoked with the argument `autoboot` and runs a `fsck` with option `-p` to ''preen'' all the disks of minor inconsistencies resulting from the last system shutdown and to check for serious inconsistencies caused by hardware or software failure. If this auto-check and repair succeeds, then the second part of `rc` is run.

The second part of `rc,` which is run after an auto-reboot succeeds and also if `rc` is invoked when a single user shell terminates, starts all the daemons on the system, preserves editor files and clears the scratch directory `/tmp.` For further information, see `init`(8). The `rc.local` command is executed immediately before any other commands after a successful `fsck.` Normally, the first commands placed in the `rc.local` file define the machine's name, using `hostname`(1), and save any possible core image that might have been generated as a result of a system crash, `savecore`(8). The latter command is included in the `rc.local` file because the directory in which core dumps are saved is usually site specific.

See Also

init(8), reboot(8), savecore(8)

Name

rdate – network date client

Syntax

/etc/rdate [**–s**] [**–v**] [*network*]

Description

The `rdate` command is invoked at boot time to reset the system date and time to the current network date and time. The `rdate` program sends a broadcast datagram packet on the specified network or on the system default network if no network is specified. The program will then wait two seconds for responses. After that time, the arithmetic median of the responses is taken.

Options

–s The system date and time will be set to the median value.

–v The time values returned by all responding hosts will be reported.

Restrictions

In order for `rdate` to determine a network time, at least one of the running hosts on the network must be running the internet time service. Machines with their own battery-backed-up time-of-day clock may not desire to set time in this manner.

See Also

date(1)

rdump (8c)

Name

rdump – file system dump across the network

Syntax

/etc/rdump [–key [*argument* ...] *filesystem*]

Description

The `rdump` command copies to magnetic tapes, disks, or a dump image file all files changed after a certain date in the *filesystem*. The command is identical in operation to `dump`(8) except the **f** key must be specified and the file supplied should be of the following form:

remote-system-name:device-or-file

The `rdump` command initiates a remote server, `/etc/rmt`, on the remote system to access the remote device or file.

Options

With the `dump` command, you specify a string of one or more of the options described below. If no options are specified, the key **9u** is assumed.

0–9 This number is the "dump level." All files that were modified since the last date stored in the file `/etc/dumpdates` for the filesystem at lesser levels will be dumped. If no date is determined by the level, the beginning of time is assumed. Thus, the level **0** causes the entire filesystem to be dumped, a level 5 is used for a weekly backup and a level 9 for a daily backup.

B The next *argument* is a number that specifies the size, in 1024-byte blocks, of a storage medium, such as a diskette or a removable disk. See the first example.

d The density of the tape, expressed in bits per inch, is taken from the next *argument*. This density is used in calculating the amount of tape used per reel. The default density is 1600 bpi.

f Place the dump on the file or device specified by the next *argument*. This file is specified as remote-system-name:device-or-file.

n Notify, by means similar to a `wall`(1) command, all users in the group "operator" when `dump` needs operator attention.

o Provides compatibility with non-ULTRIX or pre-ULTRIX Version 2.0 remote systems.

S Displays amount of space used by `dump` without performing the dump operation. This is used for presizing either for file preallocation or to ensure the correct number of tapes or disks are on hand.

s The next *argument* specifies the size of the dump tape (in feet). When the specified size is reached, `dump` waits for the reel to be changed. The default tape size is 2300 feet.

u If the dump completes successfully, writes the date of the beginning of the dump to file `/etc/dumpdates`. This file records a separate date for each filesystem and each dump level. The format of `/etc/dumpdates` consists of one free format record per line: filesystem name, increment level and `ctime`(3) format dump date. The superuser can carefully edit `/etc/dumpdates` to change any of the fields.

W The `dump` utility tells the operator what file systems need to be dumped. This information is taken from the files `/etc/dumpdates` and `/etc/fstab`. The `W` option causes `dump` to print out, for each file system in `/etc/dumpdates`, the most recent dump date and level, and highlights those file systems that should be dumped. If the `W` option is used, all other options are ignored, and `dump` exits immediately.

w Unlike `W`, `w` lists only those filesystems that need to be dumped.

Examples

This example reports number of bytes to be output for a level 0 dump of the root file system. Please note: the file `test` is not created.

```
rdump OSf system:test /
```

This example dumps the root(/) file system from the local system to a remote system named ''nihil''. A level 0 dump to tape is performed and the `/etc/dumpdates` file is updated.

```
rdump 0uf nihil:/dev/rmt0h /
```

This example dumps the user (/usr) file system from the local system to a remote system named ''dickens'' using the RX50 device named `/dev/rra1a` for output. A level 0 dump is performed, but the `/etc/dumpdates` file is not updated.

```
rdump 0f dickens:/dev/rra1a /usr
```

This example dumps the root file system to a non-ULTRIX or ULTRIX before Version 2.0 remote system. A level 0 dump to a tape drive is performed and the `/etc/dumpdates` file is updated.

```
rdump 0ouf system:/dev/rmt0h /
```

This example dumps the root file system to a non-ULTRIX or ULTRIX before Version 2.0 remote system. A level 0 dump to the rx50 device is performed and the `/etc/dumpdates` file is updated. Note the specification of 400 as the device size in 1,024 byte blocks.

```
rdump 0ouBf 400 system:/dev/rra1a /
```

If you want to use / and `/usr`, you must have these entries in the `/etc/fstab` file.

Diagnostics

Same as `dump`(8) with a few extra related to the network.

Files

`/dev/tt`	Required for user interface

See Also

dump(8), rmt(8c)

Name

reboot – automatic reboot procedures

Syntax

/etc/reboot [**-n**] [**-q**]

Description

The ULTRIX system is booted by loading a kernel image, usually `/vmunix`, into memory at location zero and transferring to zero. Because the system is not reenterable, the kernel image must be read in from disk each time the system is bootstrapped.

When the reboot of a running system is desired, `shutdown` is normally used. If there are no users, `/etc/reboot` can be used. The `reboot` command causes the disks to be synced, and then a multiuser reboot is initiated. The system is booted and an automatic disk check is performed. If the procedure succeeds, the system is then brought up for the users.

The system will reboot itself after a power failure or after a crash, provided auto-restart is enabled on your system. A consistency check of the file systems will be performed and, unless the check fails, the system will resume multiuser operations.

Options

-n	Prevents the disks from being synced.
-q	Reboots quickly and ungracefully, without shutting down running processes first.

Files

`/vmunix`	System code

See Also

crash(8v), fsck(8), halt(8), init(8), newfs(8), rc(8), shutdown(8)

remnode(8)

Name

remnode – remove one or more entries from the nodes database

Syntax

/etc/remnode [*node*]

Description

For each given *node* argument, `remnode` deletes the corresponding node entry (or entries) from the nodes database. The nodes database is the one used by DECnet. The *node* is either the node address or the node name for each node entry that you want to delete. Note, you can specify more than one *node* argument in a single command.

A node address is a decimal integer in the range of 1 to 1023 for single area networks, or has the format *a.n* for multiarea networks, where *a* is the network area number (a decimal integer in the range of 2 to 63) and *n* is the node number (a decimal integer in the range of 1 to 1023).

A node name can be from one to six alphanumeric characters, including at least one letter.

Examples

This command removes the entries for nodes 44.70 and Mynode from the nodes database:

```
# /etc/remnode 44.70 mynode <RET>
```

This command removes the entry for node lttwi from the nodes database:

```
# /etc/remnode lttwi <RET>
```

See Also

addnode(8), ccr(8), getnode(8), load(8), mop_mom(8), trigger(8)

Name

renice – alter priority of running processes

Syntax

/etc/renice priority [[**-p**] pid ...] [[**-g**] pgrp ...] [[**-u**] user ...]

Description

The `renice` command alters the scheduling priority of one or more running processes. The *who* parameters are interpreted as process ID's, process group ID's, or user names. Using `renice` on a process group causes all processes in the process group to have their scheduling priority altered. Using `renice` on a user causes all processes owned by the user to have their scheduling priority altered. By default, the processes to be affected are specified by their process ID's.

Options

To force *who* parameters to be interpreted as process group ID's, a `-g` may be specified. To force the *who* parameters to be interpreted as user names, a `-u` may be given. Supplying `-p` will reset *who* interpretation to be (the default) process ID's.

Users other than the superuser may only alter the priority of processes they own, and can only monotonically increase their ''nice value'' within the range 0 to PRIO_MIN (20). (This prevents overriding administrative fiats.) The superuser can alter the priority of any process and set the priority to any value in the range PRIO_MAX (–20) to PRIO_MIN. Useful priorities are: 19 (the affected processes will run only when nothing else in the system wants to), 0 (the ''base'' scheduling priority), anything negative (to make things go very fast).

Examples

The following command changes the priority of process ID's 987 and 32, and all processes owned by users daemon and root:

```
/etc/renice +1 987 -u daemon root -p 32
```

Restrictions

If you make the priority very negative, then the process cannot be interrupted. To regain control you make the priority greater than zero. Non-superusers cannot increase scheduling priorities of their own processes, even if they were the ones that decreased the priorities in the first place.

Files

`/etc/passwd` Maps user names to user IDs

See Also

getpriority(2), setpriority(2)

Name

repquota – summarize quotas for a file system

Syntax

repquota *filesys*...

Description

The `repquota` command prints a summary of the disc usage and quotas for the specified file systems. For each user the current number files and amount of space (in kilobytes) is printed, along with any quotas created with `edquota`(8).

Only superusers can view quotas that are not their own.

Diagnostics

Various messages about inaccessible files; self-explanatory.

Files

quotas	At the root of each file system with quotas
`/etc/fstab`	For file system names and locations

See Also

quota(1), quota(2), edquota(8), quotacheck(8), quotaon(8)
''Disk Quotas in a UNIX environment,'' *ULTRIX Supplementary Documents, Volume 3: System Manager*

Name

restore – incremental file system restore

Syntax

/etc/restore *key* [*name...*]

Description

The `restore` command reads from magnetic tapes, disks, a file, or a pipe created by the `dump(8)` command. The default dump media from which files are read is `/dev/rmt0h`. You can request another dump device or dump image file by using the **f** key modifier. The *key* is a character string containing one function letter and possibly one or more function modifiers. Other arguments to the command are file or directory names specifying the files to be restored. Unless the **h** key is specified, the appearance of a directory name refers to all files and, recursively, the subdirectories of that directory. The function portion of the key is specified by one of the following letters:

Keys

i This key allows interactive restoration of files from the dump media. After reading in the directory information from the dump media, `restore` lets the user move around the directory tree selecting or deselecting files to be extracted. The available interactive commands are:

ls [*arg*] – List the specified directory. If no directory is specified, the user's current directory is listed. Entries that are directories are appended with a slash (/). Entries that have been marked for extraction are prepended with an asterisk (*). If the **verbose** key is set, the inode number of each entry is also listed.

cd *arg* – Change the current working directory to the directory specified.

pwd – Print the full pathname of the current working directory.

add [*arg*] – The current directory or the specified argument (a directory or file) is added to the extraction list (the list of files to be extracted). If a directory is specified, then it and all its descendents are added to the extraction list, unless the **h** key is specified on the command line. Files that are on the extraction list are prepended with an asterisk (*) when they are listed by **ls** .

delete [*arg*] – The current directory or specified argument is deleted from the extraction list (the list of files to be extracted). If a directory is specified, then it and all its descendents are deleted from the extraction list, unless the **h** key modifier is specified on the command line. The easiest way to extract most of the files from a directory is to add the directory to the extraction list and then delete those files that are not needed.

extract – All the files on the extraction list are extracted from the dump media. The `restore` command asks which volume the user wishes to mount.

verbose – The verbose (**v**) key is toggled. Entering the command turns on verbose. Entering the command again turns off verbose. When used, the verbose key causes the **ls** command to list the inode numbers of all entries. It also causes `restore` to print out information about each file as it is extracted.

help – List a summary of the available commands.

quit – The `restore` utility immediately exits, even if the extraction list is not empty.

R The `restore` utility prompts for a particular volume of a multivolume set on which to restart a full restore. This option lets `restore` be interrupted and then restarted.

r The dump media's data is read into the current directory. You should use this function key only to restore the complete dump media onto a newly created file system, or to restore incremental dump media after a full level-0 restore. See the Examples section for a typical sequence to restore complete dump media. Note that `restore` leaves a file, `restoresymtab`, in the root directory to pass information between incremental restore passes. Remove this file after the last incremental dump media has been restored. A `dump`(8) followed by a `newfs`(8) and a `restore`(8) can be used to change the size of a file system.

t The names of the specified files are listed if they occur on the dump media. If no *name* argument is given, then the root directory is listed. This results in the entire contents of the dump media being listed, unless the **h** key modifier has been specified.

x The files specified by the *name* argument are extracted from the dump media. If a named file matches a directory whose contents had been written onto the dump media and the **h** key modifier is not specified, the directory is recursively extracted. The owner, modification time, and mode are restored, if possible. If no *name* argument is given, the root directory is extracted. This results in the extraction of the entire contents of the dump media.unless the **h** key modifier has been specified.

You can use any of the following characters in addition to the letter that selects the function desired:

B The next argument to `restore` is a number giving the size, in 1024-byte blocks, of a fixed-size storage medium, such as diskettes or removable disks (see the Examples section). The `restore` command does not ask whether it should abort the restore if there is a dump media read error. It always tries to skip over the bad block(s) and continue.

f The next argument to `restore` is used as the name of the archive instead of `/dev/rmt0h`. If the argument is a dash (–), `restore` reads from standard input (see the Examples section).

h The `restore` command extracts the actual directory, rather than the files that it references. This prevents hierarchical restoration of complete subtrees from the dump media:

m The `restore` command extracts by inode numbers rather than by file name. This is useful if only a few files are being extracted, and you want to avoid typing the complete pathname to the file.

s The next argument identifies which dump file on the dump media is to be used by `restore`. This is useful when the dump media has more than one dump image on it and not all of them will be restored.

v Normally, `restore` does its work silently. The **v** (verbose) key modifier causes it to display the name of each file it treats, preceded by its file type.

Examples

The following example shows a typical sequence of commands to restore complete dump media.

```
/etc/newfs /dev/rra0g ra60
/etc/mount /dev/ra0g /mnt
cd /mnt
restore r
```

Another `restore` can be done to get an incremental dump.

The following example shows how `dump(8)` and `restore(8)` can be used in a pipeline to dump and restore a file system:

```
dump 0f - /usr | (cd /mnt; restore xf -)
```

The following example shows how to restore files interactively from a dump on RX50 diskettes:

```
restore iBf 400 /dev/ra2a
```

Restrictions

The `restore` utility can make errors when doing incremental restores from dump media that were made on active file systems.

You must do a level 0 dump after a full restore. Because `restore` runs in user code, it has no control over inode allocation; thus, you must do a full `restore` to get a new set of directories that reflects the new inode numbering, even though the contents of the files are unchanged.

Diagnostics

Complains about bad key characters.

Complains if it gets a dump media read error. If the user responds with a y, `restore` attempts to continue the restore.

If the dump extends over more than one dump volume, `restore` will ask the user to change volumes. If the **x** or **i** function key has been specified, `restore` also asks which volume the user wishes to mount.

There are numerous consistency checks that can be listed by `restore`. Most checks are self-explanatory. Some common errors are:

Converting to new file system format
If dump media created from the Fast File System (FFS) has been loaded. It is automatically converted to the Berkeley Version 4.2 file system format.

***<filename>*: not found on tape{disk}**
The specified file name was listed in the dump media directory, but was not found on the media. This is caused by dump media read errors while looking for the file or from using dump media created on an active file system. **Expected next file *<inumber>*, got *<inumber>***
A file that was not listed in the directory was found on the media. This can occur when using dump media created on an active file system.

Incremental tape{disk} too low
When doing incremental restore, dump media was loaded that was written before the previous incremental media or has too low an incremental level.

Incremental tape{disk} too high
When doing incremental restore, dump media that does not begin its coverage where the previous incremental dump media left off, or that has too high an incremental level has been loaded.

Tape{Disk} read error while restoring *<filename>*
Dump media read error while skipping over inode *<inumber>*
Dump media read error while trying to resynchronize
A dump media read error has occurred. If a file name is specified, then its contents are probably partially wrong. If an inode is being skipped or the restore is trying to resynchronize, then no extracted files have been corrupted, although files may not be found on the dump media.

resync restore, skipped <num> blocks
After a dump media read error, `restore` may have to resynchronize itself. This message lists the number of blocks that were skipped.

Files

`/dev/rmt0h`	Default tape drive
`/tmp/rstdir`	File containing directories on the dump media
`/tmp/rstmode*`	Owner, mode, and time stamps for directories
`/restoresymtab`	Information passed between incremental restores
`/dev/tty`	Required for user interface

See Also

dump(8), mkfs(8), mount(8), rrestore(8c)

Name

rexecd – remote execution server

Syntax

/etc/rexecd

Description

The `rexecd` command is the server for the `rexec`(3x) routine. The server provides remote execution facilities with authentication based on usernames and encrypted passwords.

The `rexecd` command is invoked by `inetd`(8c) when it receives a connection on the port indicated in the ''exec'' service specification. For further information, see `services`(5). When a service request is received the following protocol is initiated:

1) The server reads characters from the socket up to a null ('\0') byte. The resultant string is interpreted as an ASCII number, base 10.

2) If the number received in step 1 is non-zero, it is interpreted as the port number of a secondary stream to be used for the **stderr**. A second connection is then created to the specified port on the client's machine.

3) A null terminated username of at most 16 characters is retrieved on the initial socket.

4) A null terminated password of at most 16 characters is retrieved on the initial socket.

5) A null terminated command to be passed to a shell is retrieved on the initial socket. The length of the command is limited by the upper bound on the size of the system's argument list.

6) The `rexecd` command then validates the user as is done at login time and, if the authentication was successful, changes to the user's home directory, and establishes the user and group protections of the user. If any of these steps fail the connection is aborted with a diagnostic message returned.

7) A null byte is returned on the connection associated with the **stderr** and the command line is passed to the normal login shell of the user. The shell inherits the network connections established by `rexecd`.

Diagnostics

All diagnostic messages are returned on the connection associated with the **stderr**, after which any network connections are closed. An error is indicated by a leading byte with a value of 1 (0 is returned in step 7 above upon successful completion of all the steps prior to the command execution).

username too long
The name is longer than 16 characters.

password too long
The password is longer than 16 characters.

command too long
The command line passed exceeds the size of the argument list (as configured into the system).

Login incorrect
No password file entry for the username existed.

Password incorrect
The wrong was password supplied.

No remote directory
The `chdir` command to the home directory failed.

Try again
A *fork* by the server failed.

/bin/sh: ...
The user's login shell could not be started.

Restrictions

Indicating ''Login incorrect'' as opposed to ''Password incorrect'' is a security breach which allows people to probe a system for users with null passwords.

See Also

inetd(8c)

Name

ris – remote installation services utility

Syntax

/etc/ris

Description

The `ris` utility performs remote installation services which install system software to a client machine through the TCP/IP local network. The client machine can be a VAX or a RISC machine.

The server on which the remote installation services area is located can be either a VAX or a RISC machine.

The remote installation services utility uses the directory `/usr/var/adm/ris` as a base. When you install the first product to `/usr/var/adm/ris,` the `ris` utility creates a remote installation services area. The area is called either `ris0.mips` or `ris0.vax.`

The area contains one or more ULTRIX software products. Each product contains the subsets of the kits that can be installed to clients on a network. Clients registered for an area install software over the network to their processor.

The `ris` utility performs the following functions:

a - Add client

r - Remove client

s - Show products in remote installation environments

m - Modify client

i - Install software

You must use the `ris` utility interactively to set up a remote installation services area. After you have set up a remote installation services area on the server, you can use the `ris` utility either interactively or from the command line to manage clients.

The `/usr/var/adm/ris` directory also holds a subdirectory, `/usr/var/adm/ris/clients.` This directory contains a database file, `risdb,` that you can use to manage multiple clients.

Examples

The example that follows invokes the `ris` utility interactively:

```
# /etc/ris
```

A menu appears from which you can select options to perform `ris` functions.

The examples that follow manage the client `bergal` using the command line.

The syntax of the command to add a client follows:

```
/etc/ris -a <clientname> -h <Ethernet_address> -p path,product[,product]
```

The following command adds client `bergal` and allows that client to install a product over the network:

```
# /etc/ris -a bergal -h 08-00-2B-03-05-8B -p ris0.mips,product_1
```

The syntax of the command to modify a client follows:

```
/etc/ris -a <clientname> [-h <Ethernet_address>] [-p <path,product,product>]
```

The following command modifies client `bergal` and allows that client to install a product over the network:

```
# /etc/ris -m bergal -p ris0.mips,product_2
```

The syntax of the command to remove a client follows:

```
/etc/ris -r <clientname>
```

The following command removes client `bergal`:

```
# /etc/ris -r bergal
```

Files

/usr/var/adm/ris

/usr/var/adm/ris/clients

/usr/var/adm/ris/clients/risdb

See Also

setld(8)
Guide to the Remote Installation Services
Guide to Server Setup

Name

rlogind – remote login server

Syntax

/etc/rlogind

Description

The `rlogind` server is used for the `rlogin`(1c) program. The server provides a remote login facility with authentication based on privileged port numbers.

The `rlogind` server is invoked by `inetd`(8c) when it receives a connection on the port indicated in the login service specification. For further information, see `services`(5). When a service request is received, the following protocol is initiated:

1. The server checks the client's source port. If the port is not in the range 0-1023, the server aborts the connection.
2. The server checks the client's source address and requests the corresponding host name. If the hostname cannot be determined, the dot-notation representation of the host address is used.

Once the source port and address have been checked, `rlogind` allocates a pseudo terminal and manipulates file descriptors so that the slave half of the pseudo terminal becomes the **stdin**, **stdout**, and **stderr** for a login process. For further information, see `pty`(4),

The login process is an instance of the `login`(1) program, invoked with the `-r` option. The login process then proceeds with the authentication process as described in `rshd`(8c), but if automatic authentication fails, it reprompts the user to log in on a standard terminal line.

The parent of the login process manipulates the master side of the pseudo terminal, operating as an intermediary between the login process and the client instance of the `rlogin` program. In normal operation, the packet protocol described in `pty`(4) is invoked to provide ^S/^Q type facilities and propagate interrupt signals to the remote programs. The login process propagates the client terminal's baud rate and terminal type, as found in the environment variable, TERM. For further information see `environ`(7).

The screen or window size of the terminal is requested from the client, and any changes in the window size from the client are sent to the pseudo terminal.

Restrictions

The authentication procedure used here assumes the integrity of each client machine and the connecting medium. This is insecure, but it is useful in an open environment.

Diagnostics

All diagnostic messages are returned on the connection associated with the **stderr**, after which any network connections are closed. An error is indicated by a leading byte with a value of 1.

Hostname for your address unknown
No entry in the host name database existed for the client's machine.

Try again
A *fork* by the server failed.

/bin/sh: ...
The user's login shell could not be started.

See Also

rlogin(1c), inetd(8c)

Name

rmt – remote mass storage device protocol module

Syntax

/etc/rmt

Description

The `rmt` program is used by the remote dump and restore programs to manipulate remote mass storage devices and files through an interprocess communication connection. The `rmt` program is normally started with an `rexec`(3x) or `rcmd`(3x) call.

The `rmt` program remotely ties its standard input and output to a socket, accepts commands that manipulate remote devices or files, performs the commands, and then responds with a status indication. All commands and responses are in ASCII and in one of two forms. Successful commands have responses of an acknowledgement (ack) such as:

A*number*\n

In this example, *number* is an ASCII representation of a decimal number. Unsuccessful commands are responded to with the following:

E*error-number*\n*error-message*\n,

In this example, *error-number* is one of the possible error numbers described in `intro`(2) and *error-message* is the corresponding error string as printed from a call to `perror`(3). The protocol is comprised of the following commands (a newline (\n) is present between each token):

O *device mode*	Open the specified *device* using the indicated *mode*. The *device* is a full pathname and *mode* is an ASCII representation of a decimal number suitable for passing to `open`(2). If a device had already been opened, it is closed before a new open is performed. Device can be a regular file.
C *device*	Close the currently open device or file. The *device* specified is ignored.
D	Returns generic device information for the open device. A DEVIOCGET `ioctl`(2) call is performed and the data returned. If the operation is successful, an ''ack'' is sent with the size of the information buffer.
L *whence offset*	Perform an `seek`(2) operation using the specified parameters. The response value is that returned from the `lseek` call.
P	Returns disk partition information of the open device. A DIOCDGTPT `ioctl`(2) call is performed and the data returned. If the operation is successful, an ''ack'' is sent with the size of the information buffer.
T *filename*	Returns file status information for the specified file. A `stat`(2) call is performed and the data returned. If the operation is successful, an ''ack'' is sent with the size of the information buffer.

W *count*	Write data onto the open device. The `rmt` program reads *count* bytes from the connection, aborting if a premature end-of-file is encountered. The response value is that returned from the `write`(2) call. If the operation was successful, an ''ack'' is sent containing the number of bytes written.
R *count*	Read *count* bytes of data from the open device. If *count* exceeds the size of the data buffer (10 kilobytes), it is truncated to the data buffer size. The `rmt` program then performs the requested `read`(2) and responds with A*count-read*\n if the read was successful. Otherwise an error in the standard format is returned. If the read was successful, the data read is then sent.
I *operation count*	Perform a MTIOCTOP `ioctl`(2) command using the specified parameters. The parameters are interpreted as the ASCII representations of the decimal values to place in the *mt_op* and *mt_count* fields of the structure used in the `ioctl` call. The return value is the *count* parameter when the operation is successful.
S	Return the status of the open device, as obtained with a MTIOCGET `ioctl` call. If the operation was successful, an ''ack'' is sent with the size of the status buffer, then the status buffer is sent (in binary).

Any other command causes `rmt` to exit.

Restrictions

Do not use `rmt` for a remote file access protocol.

Diagnostics

All responses are of the form described above.

See Also

rcmd(3x), rexec(3x), mtio(4), rdump(8c), rrestore(8c)

Name

route – manually manipulate the routing tables

Syntax

/etc/route [**–f**] [**–n**] *command args*]

Description

The `route` program is used to manipulate the network routing tables manually. However, normally it is not needed, as the system routing table management daemon, `routed`(8c), should tend to this task.

The `route` program accepts two commands: *add*, to add a route and *delete*, to delete a route.

All commands have the following syntax:

> **/etc/route** *command* [**net** | **host**] *destination gateway* [*metric*]

In this syntax, *destination* is a host or network for which the route is to, *gateway* is the gateway to which packets should be addressed, and *metric* is an optional count indicating the number of hops to the *destination*. The metric is required for **add** commands. It must be zero if the destination is on a directly-attached network, and nonzero if the route utilizes one or more gateways.

When adding a route with metric 0, the gateway given is the address of this host on the common network, indicating the interface to be used for transmission. Routes to a particular host are distinguished from routes to a network by interpreting the Internet address associated with *destination*. The optional keywords **net** and **host** force the destination to be interpreted as a network or host, respectively. If the *destination* has a local address part of INADDR_ANY, then the route is assumed to be to a network. Otherwise, it is presumed to be a route to a host. If the route is to a destination connected via a gateway, the *metric* should be greater than 0. All symbolic names specified for a *destination* or *gateway* are looked up first as a host name using `gethostbyname`(3n.) If this lookup fails, `getnetbyname`(3n) is then used to interpret the name as that of a network.

The `route` command uses a raw socket and the SIOCADDRT and SIOCDELRT ioctls to do its work. As such, only the superuser can modify the routing tables.

Options

–f Flushes the routing tables of all gateway entries. If `-f` is used with one of the commands described above, the tables are flushed prior to the command's application.

–n Prevents attempts to print host and network names symbolically when reporting actions.

Restrictions

The change operation is not implemented. Therefore, you should first add the new route, and then delete the old one.

Diagnostics

add [host | network] %s: gateway
The specified route is being added to the tables. The values printed are from the routing table entry supplied in the *ioctl* call. If the gateway address used was not the primary address of the gateway (the first one returned by `gethostbyname)`, the gateway address is printed numerically as well as symbolically.

delete [host | network] %s: gateway %s flags %x
The specified route is being deleted from the tables. The values printed are from the routing table entry supplied in the *ioctl* call. If the gateway address used was not the primary address of the gateway (the first one returned by `gethostbyname)`, the gateway address is printed numerically as well as symbolically.

%s %s done
When the `-f` flag is specified, each routing table entry that is deleted is indicated with a message of this form.

Network is unreachable
An attempt to add a route failed because the gateway listed was not on a directly-connected network. The next-hop gateway must be given.

not in table
A delete operation was attempted for an entry which was not present in the tables.

routing table overflow
An add operation was attempted, but the system was low on resources and was unable to allocate memory to create the new entry.

See Also

intro(4n), routed(8c)

Name

routed – network routing daemon

Syntax

/etc/routed [*options*] [*logfile*]

Description

The `routed` program is invoked at boot time to manage the network routing tables. The routing daemon uses a variant of the Xerox NS Routing Information Protocol in maintaining up-to-date kernel routing table entries.

In normal operation the `routed` program listens on a `udp(4p)` socket for packets of routing information. If the host is an internetwork router, it periodically supplies copies of its routing tables to any directly connected hosts and networks.

When `routed` is started, it uses the SIOCGIFCONF *ioctl* to find those directly connected interfaces configured into the system and marked up (the software loopback interface is ignored). If multiple interfaces are present, it is assumed that the host will forward packets between networks. The `routed` command then transmits a request packet on each interface using a broadcast packet, if the interface supports it, and enters a loop, listening for request and response packets from other hosts.

When a request packet is received, `routed` formulates a reply based on the information maintained in its internal tables. The response packet generated contains a list of known routes, each marked with a hop count metric. A count of 16 or greater is considered infinite. The metric associated with each route returned provides a metric 'relative to the sender'.

The response packets received by `routed` are used to update the routing tables if one of the following conditions is satisfied:

- No routing table entry exists for the destination network or host, and the metric indicates the destination is reachable. That is, the hop count is not infinite.
- The source host of the packet is the same as the router in the existing routing table entry. That is, updated information is being received from the very internetwork router through which packets for the destination are being routed.
- The existing entry in the routing table has not been updated for some time (defined to be 90 seconds) and the route is at least as cost effective as the current route.
- The new route describes a shorter route to the destination than the one currently stored in the routing tables. The metric of the new route is compared against the one stored in the table to decide this.

When an update is applied, the `routed` command records the change in its internal tables and generates a response packet to all directly connected hosts and networks. The `routed` command waits a short period of time (no more than 30 seconds) before modifying the kernel's routing tables to allow possible unstable situations to settle.

In addition to processing incoming packets, the `routed` command periodically checks the routing table entries. If an entry has not been updated for 3 minutes, the entry's metric is set to infinity and marked for deletion. Deletions are delayed an additional 60 seconds to insure the invalidation is propagated throughout the internet.

Hosts acting as internetwork routers supply their routing tables every 30 seconds to all directly connected hosts and networks. The response is sent to the broadcast address on nets capable of that function, to the destination address on point-to-point links, and to the router's own address on other networks. The normal routing tables are bypassed when sending responses. The reception of responses on each network is used to determine if that the network and interface are functioning correctly. If no response is received on an interface, another route may be chosen to route around the interface, or the route may be dropped if no alternative is available.

The `routed` program supports the notion of distant passive and active gateways. When `routed` is started up, it reads the file `/etc/gateways` to find gateways which may not be identified using the SIOGIFCONF *ioctl*. Gateways specified in this manner should be marked passive if they are not expected to exchange routing information, while gateways marked active should be willing to exchange routing information (that is, they should have a `routed` process running on the machine). Passive gateways are maintained in the routing tables forever, and information regarding their existence is included in any routing information transmitted.

Active gateways are treated equally to network interfaces. Routing information is distributed to the gateway and if no routing information is received for a period of time, the associated route is deleted.

External gateways are also passive, but are not placed in the kernel routing table nor are they included in routing updates. The function of external entries is to inform `routed` that another routing process will install such a route, and that alternate routes to that destination should not be installed. Such entries are only required when both routers may learn of routes to the same destination.

The `/etc/gateways` is a series of lines, each in the following format:

< **net** | **host** > *name1* **gateway** *name2* **metric** *value* < **passive** | **active** | **external** >

The **net** or **host** keyword indicates if the route is to a network or specific host.

The *name1* is the name of the destination network or host. This may be a symbolic name located in `/etc/networks` or `/etc/hosts,` or an Internet address specified in dot notation. For further information, see `inet`(3n).

The *name2* is the name or address of the gateway to which messages should be forwarded.

The *value* is a metric indicating the hop count to the destination host or network.

The keywords **passive, active**, or **external** indicate if the gateway should be treated as passive or active (as previously described), or whether the gateway is external to the scope of the `routed` protocol.

Any other argument supplied is interpreted as the name of a file in which the actions of `routed` should be logged. This log contains information about any changes to the routing tables and a history of recent messages sent and received which are related to the changed route.

Options

–d Enables additional debugging information to be logged, such as bad packets received.

–g Offers a route, on internetwork routers, to the default destination. This is typically used on a gateway to the Internet, or on a gateway that uses another routing protocol whose routes are not reported to other local routers.

–s Forces `routed` to supply routing information whether it is acting as an internetwork router or not.

–q Opposite of the `-s` option.

–t Prints all packets, sent or received, on the standard output. In addition, `routed` continues to receive input from the controlling terminal, so that interrupts from the keyboard will kill the process.

Restrictions

The kernel's routing tables may not correspond to those of `routed` for short periods of time while processes utilizing existing routes exit; the only remedy for this is to place the routing process in the kernel.

The `routed` command should listen to intelligent interfaces, such as an IMP, and to error protocols, such as ICMP, to gather more information. However, it does not always detect unidirectional failures in network interfaces, such as when the output side fails.

Files

`/etc/gateways`
For distant gateways

See Also

udp(4p), htable(8)

rpcinfo (8nfs)

Name

rpcinfo – report remote procedure call (RPC) information

Syntax

/etc/rpcinfo −p [*host*]
/etc/rpcinfo −u *host program-number* [*version-number*]
/etc/rpcinfo −t *host program-number* [*version-number*]

Description

The `rpcinfo` command makes a remote procedure call (RPC) call to an RPC server and displays its findings based on the specified options. The *program-number* argument can be either a name or a number. If no version is given, it defaults to 1.

Options

−p	Probe the portmapper `portmap`(8nfs), running on *host,* and print a list of all registered RPC programs. If *host* is not specified, `rpcinfo` defaults to the value returned by either `hostname`(1) or `gethostname`(2).
−u	Make an RPC call to procedure 0 of *program-number* using the user datagram protocol (UDP), and report whether a response was received.
−t	Make an RPC call to procedure 0 of *program-number* using the transmission control protocol (TCP), and report whether a response was received.

Files

`/etc/rpc`	Names for RPC program numbers

See Also

hostname(1), tcp(4p), udp(4p), portmap(8nfs)

Name

rrestore – restore a file system dump across the network

Syntax

/etc/rrestore [*–key* [*name* ...]]

Description

The `rrestore` utility obtains files from a file, magnetic tape, or disk, which were saved by a previous `dump`(8). The utility is identical in operation to `restore`(8) except the **f** key must be specified and the file supplied should be of the following form:

remote-system-name:device-or-file

The `rrestore` command initiates a remote server, `/etc/rmt,` on the remote machine to access the remote file or device. The function portion of the key is specified by one of the following letters:

Keys

i This key allows interactive restoration of files from the dump media. After reading in the directory information from the dump media, `rrestore` lets the user move around the directory tree selecting or deselecting files to be extracted. The available interactive commands are:

ls [*arg*] – List the specified directory. If no directory is specified, the user's current directory is listed. Entries that are directories are appended with a slash (/). Entries that have been marked for extraction are prepended with an asterisk (*). If the **verbose** key is set, the inode number of each entry is also listed.

cd *arg* – Change the current working directory to the directory specified.

pwd – Print the full pathname of the current working directory.

add [*arg*] – The current directory or the specified argument (a directory or file) is added to the extraction list (the list of files to be extracted). If a directory is specified, then it and all its descendents are added to the extraction list, unless the **h** key is specified on the command line. Files that are on the extraction list are prepended with an asterisk (*) when they are listed by **ls** .

delete [*arg*] – The current directory or specified argument is deleted from the extraction list (the list of files to be extracted). If a directory is specified, then it and all its descendents are deleted from the extraction list, unless the **h** key modifier is specified on the command line. The easiest way to extract most of the files from a directory is to add the directory to the extraction list and then delete those files that are not needed.

extract – All the files on the extraction list are extracted from the dump media.

The `rrestore` command asks which volume the user wishes to mount.

verbose – The verbose (**v**) key is toggled. Entering the command turns on verbose. Entering the command again turns off verbose. When used, the verbose key causes the **ls** command to list the inode numbers of all entries. It also causes `rrestore` to print out information about each file as it is extracted.

help – List a summary of the available commands.

quit – The `rrestore` utility immediately exits, even if the extraction list is not empty.

R The `rrestore` utility prompts for a particular volume of a multivolume set on which to restart a full restore. This option lets `rrestore` be interrupted and then restarted.

r The dump medias' data is read into the current directory. You should use this function key only to restore the complete dump media onto a newly created file system, or to restore incremental dump media after a full level-0 restore. See Examples for a typical sequence to restore complete dump media. Note that `rrestore` leaves a file, `restoresymtab`, in the root directory to pass information between incremental restore passes. Remove this file after the last incremental dump media has been restored. A `rdump`(8) followed by a `newfs`(8) and a `rrestore`(8) can be used to change the size of a file system.

t The names of the specified files are listed if they occur on the dump media. If no *name* argument is given, then the root directory is listed. This results in the entire contents of the dump media being listed, unless the **h** key modifier has been specified.

x The files specified by the *name* argument are extracted from the dump media. If a named file matches a directory whose contents had been written onto the dump media and the **h** key modifier is not specified, the directory is recursively extracted. The owner, modification time, and mode are restored, if possible. If no *name* argument is given, the root directory is extracted. This results in the extraction of the entire contents of the dump media unless the **h** key modifier has been specified.

You can use any of the following characters in addition to the letter that selects the function desired:

B The next argument to `rrestore` is a number giving the size, in 1024-byte blocks, of a fixed-size storage medium, such as diskettes or removable disks (see Examples). The `rrestore` utility does not ask whether it should abort the restore if there is a dump media read error. It always tries to skip over the bad block(s) and continue.

f The next argument to `rrestore` is used as the name of the remote system followed by a colon and the device or file containing the dump data.

h The `rrestore` utility extracts the actual directory, rather than the files that it references. This prevents hierarchical restoration of complete subtrees from the dump media.

m	The `rrestore` utility extracts by inode numbers rather than by file name. This is useful if only a few files are being extracted, and you want to avoid typing the complete pathname to the file.
o	Provides compatibility with non-ULTRIX or pre-ULTRIX V2.0 remote systems.
s	The next argument identifies which dump file on the dump media is to be used by `rrestore`. This is useful when the dump media has more than one dump image on it and not all of them will be restored.
v	Normally, `rrestore` does its work silently. The **v** (verbose) key modifier causes it to display the name of each file it treats, preceded by its file type.

Examples

The following example shows a typical sequence of commands to restore complete dump media from a system named ''remotesystem'' mounted on a tape device on that system:

```
/etc/newfs /dev/rrp0g ra60
/etc/mount /dev/rp0g /mnt
cd /mnt
rrestore rf remotesystem:/dev/rmt0h
```

Another `rrestore` can be done to get an incremental dump. The following example shows how to restore files interactively from a dump on RX50 diskettes:

```
rrestore iBf 400 remotesystem:/dev/ra2a
```

The following example restores a previously dumped file system from a - non-ULTRIX or an ULTRIX remote system prior to Version 2.0. The command restores the complete file system in verbose mode specifying a 400 block device size from a remote system's RX50 device:

```
rrestore rvoBf 400 remotesystem:/dev/ra2a
```

Restrictions

The `rrestore` utility can make errors when doing incremental restores from dump media that were made on active file systems.

You must do a level 0 dump after a full restore. Because `rrestore` runs in user code, it has no control over inode allocation; thus, you must do a full `restore` to get a new set of directories that reflects the new inode numbering, even though the contents of the files are unchanged.

Diagnostics

Complains about bad key characters.

Complains if it gets a dump media read error. If the user responds with a y, `rrestore` attempts to continue the restore.

If the dump extends over more than one dump volume, `rrestore` will ask the user to change volumes. If the **x** or **i** function key has been specified, `rrestore` also asks which volume the user wishes to mount.

There are numerous consistency checks that can be listed by `rrestore`. Most checks are self-explanatory. Some common errors are:

Converting to new file system format
If dump media created from the Fast File System (FFS) has been loaded. It is automatically converted to the Berkeley Version 4.2 file system format.

***<filename>*: not found on tape{disk}**
The specified file name was listed in the dump media directory, but was not found on the media. This is caused by dump media read errors while looking for the file or from using dump media created on an active file system.

expected next file *<inumber>*, got *<inumber>*
A file that was not listed in the directory was found on the media. This can occur when using dump media created on an active file system.

Incremental tape{disk} too low
When doing incremental restore, dump media was loaded that was written before the previous incremental media or has too low an incremental level.

Incremental tape{disk} too high
When doing incremental restore, dump media that does not begin its coverage where the previous incremental dump media left off, or that has too high an incremental level has been loaded.

Tape{Disk} read error while restoring <filename>
Dump media read error while skipping over inode <inumber>
Dump media read error while trying to resynchronize
A dump media read error has occurred. If a file name is specified, then its contents are probably partially wrong. If an inode is being skipped or the restore is trying to resynchronize, then no extracted files have been corrupted, although files may not be found on the dump media.

resync restore, skipped <num> blocks
After a dump media read error, `rrestore` may have to resynchronize itself. This message lists the number of blocks that were skipped.

Files

`/dev/rmt0h`	Default tape drive
`/tmp/rstdir*`	File containing directories on the dump media
`/tmp/rstmode*`	Owner, mode, and time stamps for directories
`/restoresymtab`	Information passed between incremental restores
`/dev/tty`	Required for user interface

See Also

restore(8), rmt(8c)

Name

rshd – remote shell server

Syntax

/etc/rshd

Description

The `rshd` command is the server for the `rcmd`(3x) routine and, consequently, for the `rsh`(1c) program. The server provides remote execution facilities with authentication based on privileged port numbers.

The `rshd` is invoked by `inetd`(8c) when it receives a connection on the port indicated in the cmd service specification. When a service request is received, the following protocol is initiated:

1) The server checks the client's source port. If the port is not in the range 0-1023, the server aborts the connection.

2) The server reads characters from the socket up to a null ('\0') byte. The resultant string is interpreted as an ASCII number, base 10.

3) If the number received in step 1 is non-zero, it is interpreted as the port number of a secondary stream to be used for the **stderr**. A second connection is then created to the specified port on the client's machine. The source port of this second connection is also in the range 0-1023.

4) The server checks the client's source address. If the address is associated with a host for which no corresponding entry exists in the host name data base the server aborts the connection. For further information, see `hosts`(5),

5) A null terminated user name of at most 16 characters is retrieved on the initial socket. This user name is interpreted as a user identity to use on the **server**'s machine.

6) A null terminated user name of at most 16 characters is retrieved on the initial socket. This user name is interpreted as the user identity on the **client**'s machine.

7) A null terminated command passed to a shell is retrieved on the initial socket. The length of the command is limited by the upper bound on the size of the system's argument list.

8) The `rshd` command validates the user according to the following steps. The remote user name is looked up in the password file and a `chdir` is performed to the user's home directory. If either the lookup or `chdir` fail, the connection is terminated. If the user is not the superuser, (user id 0), the file `/etc/hosts.equiv` or `/etc/hosts.lpd` is consulted for a list of hosts considered equivalent. If the client's host name is in this file, the authentication is considered successful. If the lookup fails, or the user is the superuser, the file `.rhosts`, in the home directory of the remote user, is checked for the machine name and identity of the user on the client's machine. If the lookup fails, the connection is terminated.

9) A null byte is returned on the connection associated with the **stderr** and the command line is passed to the normal login shell of the user. The shell inherits the network connections established by `rshd`.

Diagnostics

All diagnostic messages are returned on the connection associated with the **stderr**, after which any network connections are closed. An error is indicated by a leading byte with a value of 1 (0 is returned in step 9 above upon successful completion of all the steps prior to the command execution).

locuser too long
The name of the user on the client's machine is longer than 16 characters.

remuser too long
The name of the user on the remote machine is longer than 16 characters.

command too long
The command line passed exceeds the size of the argument list (as configured into the system).

Hostname for your address unknown.
No entry in the host name database existed for the client's machine.

Login incorrect.
No password file entry for the user name existed.

No remote directory.
The `chdir` command to the home directory failed.

Permission denied.
The authentication procedure described above failed.

Can't make pipe.
The pipe needed for the **stderr** was not created.

Try again.
A *fork* by the server failed.

/bin/sh: ...
The user's login shell could not be started.

Restrictions

The authentication procedure used here assumes the integrity of each client machine and the connecting medium. This is insecure, but is useful in an open environment.

See Also

rsh(1c), rcmd(3x), services(5), inetd(8c)

Name

rwalld – network rwall server

Syntax

/usr/etc/rpc.rwalld

Description

The `rwalld` daemon is a server that handles `shutdown`(8) requests. It is implemented by calling `wall`(1) to all the appropriate network machines. The `rwalld` daemon is normally invoked by `inetd`(8c.)

See Also

wall(1), services(5), inetd(8c), shutdown(8)

Name

rwhod – system status server

Syntax

/etc/rwhod [-b] [-l]

Description

The rwhod command is the server which maintains the database used by the rwho(1c) and ruptime(1c) programs. Its operation is predicated on the ability to broadcast messages on a network.

The rwhod command operates as both a producer and consumer of status information. As a producer of information it periodically queries the state of the system and constructs status messages which are broadcast on a network.

As a consumer of information, rwhod listens for the status messages of the other rwhod servers, validating them, then recording them in a collection of files located in the directory /usr/spool/rwho.

The rwho server transmits and receives messages at the port indicated in the rwho service specification. For more information, see services(5). The messages sent and received, take the following form:

```
struct  outmp {
        char    out_line[8];        /* tty name */
        char    out_name[8];        /* user id */
        long    out_time;           /* time on */
};
struct  whod {
        char    wd_vers;
        char    wd_type;
        char    wd_fill[2];
        int     wd_sendtime;
        int     wd_recvtime;
        char    wd_hostname[32];
        int     wd_loadav[3];
        int     wd_boottime;
        struct  whoent {
                struct  outmp we_utmp;
                int     we_idle;
        } wd_we[1024 / sizeof (struct whoent)];
};
```

All fields are converted to network byte order prior to transmission. The load averages are calculated by the w(1) program and represent load averages over the 5, 10, and 15 minute intervals prior to a server's transmission. They are multiplied by 100 for representation as an integer. The host name included is the name returned by the gethostname(2) system call, with any trailing domain name omitted. The array at the end of the message contains information about the users logged in to the sending machine. This information includes the contents of the utmp(5) entry for each active terminal line and a value indicating the time since a character was last received on the terminal line.

Messages received by the rwho server are discarded unless they originated at a rwho server's port. In addition, if the host's name, as specified in the message, contains any unprintable ASCII characters, the message is discarded. Valid messages received by `rwhod` are placed in files named `whod.fIhostname` in the directory `/usr/spool/rwho`. These files contain only the most recent message in the format previously described.

Status messages are generated approximately once every 3 minutes. The `rwhod` command performs an `nlist`(3) on `/vmunix` every 30 minutes to guard against the possibility that this file is not the system image currently operating.

Options

-b Sets the broadcast only mode. Sends outgoing `rwho` packets, but ignores incoming ones.

-l Sets the listen only mode. Collects incoming `rwho` packets from the network, but does not broadcast `rwho` data.

Restrictions

Because the `rwhod` daemon sends its information in broadcast packets it generates a large amount of network traffic. On large networks the extra traffic may be objectionable. Therefore, the `rwhod` daemon is disabled by default. To make use of the `rwhod` daemon for both the local and remote hosts, remove the comment symbols (#) from in front of the lines specifying `rwhod` in the `/etc/rc` file.

If the `rwhod` daemon is not running on a remote machine, the machine may incorrectly appear to be down when you use the `ruptime` command to determine its status. See the `ruptime`(1c) reference page for more information.

If a system has more than 40 users logged in at once, the number of users displayed by the `ruptime`(1c) and `rwho`(1c) commands is incorrect. Users who login after the fortieth user, will fail to appear in the output of the `ruptime`(1c) and `rwho`(1c) commands. This is because the maximum size limit of an Ethernet packet is 1500 bytes, and the `rwhod` daemon must broadcast its information in a single packet.

See Also

ruptime(1c), rwho(1c)

rxformat (8v)

Name

rxformat – format floppy disks

Syntax

/etc/rxformat [**–d**] *special*

Description

The `rxformat` program formats a diskette in the specified drive associated with the special device *special*. (The *special* is normally /dev/rrx0, for drive 0, or /dev/rrx1, for drive 1.) By default, the diskette is formatted single density; a `-d` flag can be supplied to force double density formatting. Single density is compatible with the IBM 3740 standard (128 bytes/sector). In double density, each sector contains 256 bytes of data.

Before formatting a diskette `rxformat` prompts for verification (this allows a user to cleanly abort the operation; note that formatting a diskette will destroy any existing data). Formatting is done by the hardware. All sectors are zero-filled.

Restrictions

A floppy may not be formatted if the header information on sector 1, track 0 has been damaged. Hence, it is not possible to format a completely degaussed disk. (This is actually a problem in the hardware.)

Diagnostics

No such device
The drive is not ready, usually because no disk is in the drive or the drive door is open.

Other error messages are self-explanatory.

Files

`/dev/rx?`

See Also

rx(4)

Name

rzdisk – SCSI disk maintenance utility

Syntax

/bin/rzdisk
/bin/rzdisk –f [*vendor* | *known*] *special*
/bin/rzdisk –h
/bin/rzdisk –r *LBN special*
/bin/rzdisk –s *LBN length special*

Description

The `rzdisk` utility, a SCSI disk maintenance program, formats a SCSI disk, scans a SCSI disk for bad blocks, and reassigns bad blocks on a SCSI disk.

The *special* file argument is a raw device pathname. When used with the `-f`, `-r`, or `-s` option, you should specify an unmounted c partition of a character device special file (for example, `/dev/rrz3c`).

Options

–f Formats a SCSI hard disk (see Restrictions).

Since Digital ships the SCSI disk already formatted, only use this option if you have encountered a serious problem and must reformat the disk. You can format a disk with the *vendor* (manufacturer) defect list or with the *known* (vendor and grown) defect list. The *grown* defect list contains any blocks that may have been reassigned during the life of the SCSI disk drive. When formatting a disk, you must specify the raw device pathname.

The following example reformats the disk on drive 3 using the known defect list:

```
/bin/rzdisk -f known /dev/rrz3c
```

The `-f` option with no additional arguments formats a SCSI floppy diskette (see Restrictions). New floppy diskettes are usually not formatted. You need to format each diskette before you can store data on it.

The following example shows how to format a floppy diskette in RX23 drive one:

```
/bin/rzdisk -f /dev/rrz1c
```

The floppy format operation is interactive. The program `rzdisk` will guide you through formatting the diskette.

–h Calls the HELP menu to the screen.

–r Reassigns a bad block on the disk (see Restrictions).

When reassigning a bad block, you must specify the *LBN*, which is a unique number (decimal notation) that represents the disk block as reported in the errorlog file, and the raw device pathname.

The following example reassigns block 222658 on the c partition of drive 3:

```
/bin/rzdisk -r 222658 /dev/rrz3c
```

The program `rzdisk` reads the specified *LBN* prior to reassigning the block. If `rzdisk` reads valid data from the block, then the block is not actually bad or the SCSI driver already reassigned the block. In this case, `rzdisk` asks if the reassignment should be canceled. Answer yes to cancel the reassignment. This prevents double reassignment and replacement of good blocks.

–s Scans for bad blocks on a specified area of the disk.

When scanning a disk, you must specify the *LBN*, which is a unique number (decimal notation) that represents the disk block relative to the start of the partition, the *length,* and the raw device pathname of the partition to scan.

To start scanning from the first block of the specified partition, use the number 0 to represent the *LBN*. When the number 0 is specified, the scan starts at the first block of the specified partition.

The *length* is a decimal number that indicates how many 512-byte blocks to scan. To scan up to and including the last block of the specified partition, use the number –1 to represent the *length*. By specifying the *length,* you define the scope of the scan within the identified partition.

The following example scans the first ten blocks of the entire disk (c partition) on drive 3:

```
/bin/rzdisk -s /dev/rrz3c 0 10
```

The following example scans the entire disk (**c** partion) on drive 3:

```
/bin/rzdisk -s /dev/rrz3c 0 -1
```

Diagnostics

The `rzdisk` program generates messages when the user is not privileged, when the *LBN* is not in the specified partition, and when the length exceeds the size of the partition.

Restrictions

You must have superuser privileges to run the `rzdisk` program.

You should not have to format your system disk.

The system should be in single-user mode and the file systems on the disk should be unmounted when running the `rzdisk` program, except when formatting floppy diskettes.

Use the `-f` option with caution and only if the SCSI hard disk drive seems corrupted. Be aware that when you format a disk, all resident data is destroyed.

Digital supports formatting, writing, and reading of High Density (HD) 3.5 inch diskettes in the RX23 disk drive.

Digital supports reading, but not formatting or writing, of Double Density (DD) 3.5 inch diskettes in the RX23 disk drive. Reliable reading of DD diskettes requires they be written only on a double density drive and have not been overwritten by an RX23 (or other) high density drive. This restriction occurs because of differences in the write heads between DD and HD drives. Data written by a DD drive cannot be completely overwritten by a HD drive.

The `-r` option is supported only with those SCSI disks that support the reassign block command.

See Also

dkio(4), rz(4), chpt(8), mount(8), restore(8), uerf(8)
Guide to the Error Logger System

Name

sa, accton – print process accounting statistics

Syntax

/etc/sa [*options*] [*file*]

/etc/accton [*file*]

Arguments

file With an argument naming an existing *file,* `accton` causes system accounting information for every process executed to be placed at the end of the file. If no argument is given, accounting is turned off.

Description

The `sa` command reports on, cleans up, and generally maintains accounting files.

The `sa` is able to condense the information in `/usr/adm/acct` into a summary file `/usr/adm/savacct,` which contains a count of the number of times each command was called and the time resources consumed. This condensation is desirable because on a large system `/usr/adm/acct` can grow by 100 blocks per day. The summary file is normally read before the accounting file, so the reports include all available information.

If a file name is given as the last argument, that file will be treated as the accounting file. The file `/usr/adm/acct` is the default.

Output fields are labeled: ''cpu'' for the sum of user+system time (in cpu seconds), ''re'' for real time (also in cpu seconds), ''k'' for cpu-time averaged core usage (in 1k units), ''avio'' for average number of I/O operations per execution. With options fields labeled ''tio'' for total I/O operations, ''k*sec'' for cpu storage integral (kilo-core seconds), ''u'' and ''s'' for user and system cpu time alone (both in cpu seconds) will sometimes appear.

Options

–a Place all command names containing unprintable characters and those used only once under the name '***other.'

–b Sort output by sum of user and system time divided by number of calls. Default sort is by sum of user and system times.

–c Besides total user, system, and real time for each command, print percentage of total time over all commands.

–d Sort by average number of disk I/O operations.

–D Print and sort by total number of disk I/O operations.

–f Force no interactive threshold compression with `–v` option.

–i Do not read in summary file.

–j Instead of total minutes for each category, give seconds per call.

–k	Sort by cpu-time average memory usage.
–K	Print and sort by cpu-storage integral.
–l	Separate system and user time; normally they are combined.
–m	Print number of processes and number of CPU minutes for each user.
–n	Sort by number of calls.
–r	Reverse order of sort.
–s	Merge accounting file into summary file `/usr/adm/savacct` when done.
–t	For each command, report ratio of real time to the sum of user and system times. If the sum of user and system times is too small to report, '*ignore*' appears in this field.
–u	Superseding all other flags, print for each command in the accounting file the user ID and command name.
–v	Followed by a number *n,* types the name of each command used *n* times or fewer. Await a reply from the terminal; if it begins with 'y', add the command to the category '**junk**.' This is used to strip out garbage.

Restrictions

Accounting is suspended when there is less than 2% free space on disk. Accounting resumes when free space rises above 4%.

Files

`/usr/adm/acct`	Raw accounting
`/usr/adm/savacct`	Summary
`/usr/adm/usracct`	Per-user summary

See Also

acct(2), ac(8)

savecore (8)

Name

savecore – save a core dump of the operating system

Syntax

/etc/savecore [*options*] *dirname* [*system*] [*corename*]

Description

The `savecore` command is meant to be called near the end of the `/etc/rc.local` file. The `savecore` command saves the core dump of the system (assuming one was made) and writes a reboot message in the shutdown log.

The `savecore` command checks the core dump to be certain it corresponds with the current running ULTRIX. If it does, it saves the core image in the file *dirname*/`vmcore.`*n* and saves the namelist in the file *dirname*/`vmunix.`*n*. The trailing *.n* in the pathnames is replaced by a number which increments each time `savecore` is run in that directory.

After saving the core and namelist images, `savecore` will save the error logger buffer into a predetermined file. The error logger buffer contains information about why the crash occurred. After `savecore` completes, the `elcsd` daemon will extract the error logger file and translate its contents into a form familiar to the `uerf`(8) program.

Before `savecore` writes out a core image, it reads a number from the file *dirname*/`minfree`. If there are fewer free blocks on the filesystem that contains *dirname* than the number obtained from the `minfree` file, a core dump is not done. If the `minfree` file does not exist, savecore always writes out the core file (assuming that a core dump was taken).

The `savecore` command also writes a reboot message in the shut down log. If the system crashed as a result of a panic, `savecore` also records the panic string in the shut down log.

For partial crash dumps, `savecore` creates a sparse core image file in *dirname*/`vmcore.`*n*. If this sparse core image file is copied or moved to another location, the file expands to its true size which can take too much file system space. Hence, to copy or move sparse core image files, you must use the `dd` command. The `dd` command has a conversion option to create sparse output files.

Options

–c Clears the core dump. This option is useful when the core dump is corrupted in a way that will not allow `savecore` to save it safely. Use the `-c` option with caution, because once it clears the core dump, the core dump cannot be retrieved.

–d *dumpdev dumplo*
Specifies the dump device and the dump offset when running `savecore` on a system image other than the currently running system image. The `savecore` program assumes that the running system image is `/vmunix` and it reads the dump device and dump device offset are different in the system image that crashed, the `-d` option provides the correct dump device and dump device offset.

–e Saves only the error logger buffer into a file. If used, core or namelist images are not saved.

–f *corename*
Takes the i *corefile* name as the file from which to extract the the crash dump data instead of the default dump device. This option is used only for diskless workstations.

If the core dump was from a system other than `/vmunix`, the name of that system must be supplied as *system*. The `savecore` program assumes that the running image is `/vmunix`.

After successful completion, the core dump is cleared. Also, a message is written in the file `/usr/adm/shutdownlog` which tells whether the dump succeeded or failed.

Files

`/usr/adm/shutdownlog`	Shut down log
`/vmunix`	Current running ULTRIX system

See Also

dd(1), uerf(8)

secsetup (8)

Name

secsetup – enable the enhanced security features

Syntax

/usr/etc/sec/secsetup

Description

The `secsetup` command is an interactive facility that allows you to enable the enhanced security features on your system. You must first have loaded the enhanced security subset onto your system before running the command.

The `secsetup` command allows you to configure your system either for security auditing, trusted path, enhanced login, or any combination of those features. In addition, the `secsetup` command may add lines to the `/etc/rc.local` file.

NOTE

To remove entries from the `/etc/rc.local` you must edit it by hand. The `secsetup` command only adds lines to this file if they aren't already present.

You can run `secsetup` while the system is in multiuser mode (however, some inconsistencies may result from this. See the *Security Guide for Administrators* for more information). To run `secsetup,` type the following and then answer the questions that follow:

```
# /usr/etc/sec/secsetup
```

Depending on the security features chosen, when `secsetup` completes you may need to replace your system's kernel and reboot the system. For example, chosing either the security auditing or trusted path feature may require you to re-build your kernel.

Files

```
/etc/sec/audit_events
/etc/auth
/etc/passwd
/etc/rc.local
/etc/svc.conf
```

See Also

set_audit_mask(8), auth(5), svc.conf(5)
Security Guide for Administrators

Name

sendmail, newaliases, mailq – send mail over the internet

Syntax

/usr/lib/sendmail [*flags*] [*address* ...]

newaliases

mailq

Description

The `sendmail` command sends a message to one or more people, routing the message over whatever networks are necessary. The `sendmail` command does internetwork forwarding as necessary to deliver the message to the correct place.

The `sendmail` command is not intended as a user interface routine. Other programs provide user-friendly front ends, while `sendmail` is used only to deliver pre-formatted messages.

With no flags, `sendmail` reads its standard input up to a CTRL/D or a line with a single dot and sends a copy of the letter found there to all of the addresses listed. It determines the network to use based on the syntax and contents of the addresses.

Local addresses are looked up in a file and aliased appropriately. Aliasing can be prevented by preceding the address with a backslash. Normally the sender is not included in any alias expansions, for example, if 'john' sends to 'group', and 'group' includes 'john' in the expansion, then the letter will not be delivered to 'john'.

Flags

–ba	Go into ARPANET mode. All input lines must end with a CR-LF, and all messages will be generated with a CR-LF at the end. Also, the 'From:' and 'Sender:' fields are examined for the name of the sender.
–bd	Run as a daemon. This requires Berkeley IPC.
–bi	Initialize the alias database.
–bm	Deliver mail in the usual way (default).
–bp	Print a listing of the queue.
–bs	Use the SMTP protocol as described in RFC 821. This flag implies all the operations of the **–ba** flag that are compatible with SMTP.
–bt	Run in address test mode. This mode reads addresses and shows the steps in parsing; it is used for debugging configuration tables.
–bv	Verify names only. Do not try to collect or deliver a message. Verify mode is normally used for validating users or mailing lists.
–bz	Create the configuration freeze file.
–C*file*	Use alternate configuration file.
–F*fullname*	Set the full name of the sender.

–f*name*	Sets the name of the from person, that is, the sender of the mail. The `-f` flag can only be used by the special users `root`, `daemon`, and `network`, or if the person you are trying to become is the same as the person you are.
–h*N*	Set the hop count to *N*. The hop count is incremented every time the mail is processed. When it reaches a limit, the mail is returned with an error message, the victim of an aliasing loop.
–n	Do not do aliasing.
–o*x value*	Set option *x* to the specified *value*. Options are described below.
–q[*time*]	Process saved messages in the queue at given intervals. If *time* is omitted, process the queue once. The *time* argument is given as a tagged number, with 's' being seconds, 'm' being minutes, 'h' being hours, 'd' being days, and 'w' being weeks. For example, '–q1h30m' or '–q90m' would both set the timeout to one hour thirty minutes.
–r*name*	An alternate and obsolete form of the `-f` flag.
–t	Read message for recipients. To:, Cc:, and Bcc: lines will be scanned for people to send to. The Bcc: line will be deleted before transmission. Any addresses in the argument list will be suppressed.
–v	Go into verbose mode. For example, alias expansions will be announced.

Options

There are also a number of processing options that may be set. Normally these will only be used by a system administrator. Options may be set either on the command line using the `-o` flag or in the configuration file.

A*file*	Use alternate alias file.
c	Do not initiate immediate connection to mailers that are considered expensive to connect to. This requires queueing.
d*x*	Set the delivery mode to *x*. Delivery modes are 'i' for interactive (synchronous) delivery, 'b' for background (asynchronous) delivery, and 'q' for queue only – that is, actual delivery is done the next time the queue is run.
D	Try to automatically rebuild the alias database if necessary.
e*x*	Set error processing to mode *x*. Valid modes are 'm' to mail back the error message, 'w' to write back the error message (or mail it back if the sender is not logged in), 'p' to print the errors on the terminal (default), 'q' to throw away error messages (only exit status is returned), and 'e' to do special processing for the BerkNet. If the text of the message is not mailed back by modes 'm' or 'w' and if the sender is local to this machine, a copy of the message is appended to the file `dead.letter` in the sender's home directory.
F*mode*	The mode to use when creating temporary files.

f	Save UNIX `From` lines at the front of messages.
g*N*	The default group id to use when calling mailers.
H*file*	The SMTP help file.
i	Do not take dots on a line by themselves as a message terminator.
L*n*	The log level.
m	Send to me (the sender) also if I am in an alias expansion.
o	If set, this message may have old style headers. If not set, this message is guaranteed to have new style headers (that is, commas instead of spaces between addresses). If set, an adaptive algorithm is used that will correctly determine the header format in most cases.
Q*queuedir*	Select the directory in which to queue messages.
r*timeout*	The timeout on reads; if none is set, `sendmail` will wait forever for a mailer.
S*file*	Save statistics in the named file.
s	Always instantiate the queue file, even under circumstances where it is not strictly necessary.
T*time*	Set the timeout on messages in the queue to the specified time. After sitting in the queue for this amount of time, they will be returned to the sender. The default is three days.
t*stz,dtz*	Set the name of the time zone.
u*N*	Set the default user id for mailers.

If the first character of the user name is a vertical bar, the rest of the user name is used as the name of a program to pipe the mail to. It may be necessary to quote the name of the user to keep `sendmail` from suppressing the blanks from between arguments.

`Sendmail` returns an exit status describing what it did. The codes are defined in <*sysexits.h*>

EX_OK	Successful completion on all addresses.
EX_NOUSER	Username not recognized.
EX_UNAVAILABLE	Catchall meaning necessary resources were not available.
EX_SYNTAX	Syntax error in address.
EX_SOFTWARE	Internal software error, including bad arguments.
EX_OSERR	Temporary operating system error, such as `cannot`fork
EX_NOHOST	Host name not recognized.
EX_TEMPFAIL	Message could not be sent immediately, but was queued.

If invoked as `newaliases`, `sendmail` will rebuild the alias database. If invoked as `mailq`, `sendmail` will print the contents of the mail queue.

Restrictions

`Sendmail` converts blanks in addresses to dots. This is incorrect according to the old ARPANET mail protocol RFC 733 (NIC 41952), but is consistent with the new protocols (RFC 822).

Files

Except for `/usr/lib/sendmail.cf`, these pathnames are all specified in `/usr/lib/sendmail.cf`. Thus, these values are only approximations.

`/etc/aliases`	Raw data for alias names
`/etc/aliases.pag`	
`/etc/aliases.dir`	Database of alias names
`/var/yp/src/mail.aliases`	Raw data for alias names
`/var/yp/DOMAINNAME/mail.aliases.pag`	Yellow Pages alias database. DOMAINNAME is the YP domainname for the local area network.
`/var/yp/DOMAINNAME/mail.aliases.dir`	Yellow Pages alias database. DOMAINNAME is the YP domainname for the local area network.
`/var/dss/namedb/src/aliases`	Raw data for alias names
`/var/dss/namedb/aliases.db`	BIND/Hesiod alias database
`/usr/lib/sendmail.cf`	Configuration file
`/usr/lib/sendmail.fc`	Frozen configuration
`/usr/lib/sendmail.hf`	Help file
`/usr/lib/sendmail.st`	Collected statistics
`/usr/bin/uux`	To deliver `uucp` mail
`/usr/lib/mailers/arpa`	To deliver ARPANET mail
`/usr/spool/mqueue/*`	Temp files

See Also

biff(1), binmail(1), mail(1), rmail(1), aliases(5), mailaddr(7)
DARPA Internet Request For Comments: RFC 819, RFC 821, RFC 822
Sendmail – An Internetwork Mail Router

Name

setld – software subset management utility

Syntax

/etc/setld [**–D** *dir*] **–l** *location* [*subset...*]

/etc/setld [**–D** *dir*] **–d** *subset...*
/etc/setld [**–D** *dir*] **-i** [*subset...*]
/etc/setld [**–D** *dir*] **–u** *location*
/etc/setld [**–D** *dir*] **–v** [*subset...*]
/etc/setld [**–D** *dir*] **–c** *subset message*

/etc/setld [**–D** *dir*] **–x** *location* [*subset...*]

Arguments

location

Specifies location of distribution. This can be be either the name of a directory, a device special file name, or the name of a remote installation server. The specified location is used to determine the type of media to be used. Valid location specifiers include the following:

/dev/rmt0h
Magnetic tape on tape unit 0

/mnt/VAX/BASE
Disk distribution in directory */mnt/VAX/BASE*

hostname:
Remote distribution from server *hostname*

subset

Specifies the name of a subset on which an operation is to be performed. Subset names are strings of seven or more characters used to uniquely identify subsets, for example, `UDTUUCP400`.

message

Specifies a string to be sent to a subset control program to configure a subset.

Function Keys

–l Load software from distribution mounted on *location*. If no optional *subset* is specified, a menu of subsets available on the distribution is presented. Any subset chosen from the menu is then loaded on the system. If an optional *subset* is specified, only that subset is loaded.

–d Delete *subset* from the system. Each *subset* named on the command line is removed from the system. All files in each *subset* which have not been modified since installation are unlinked.

Subsets can be marked by a vendor during manufacture so that they cannot be deleted. Attempts to delete such subsets will generate an appropriate diagnostic.

If a subset being deleted is required by other subsets installed to the system, these are listed and the user is asked to confirm that the subset is to be deleted.

–i Inventory the system or any specified *subset*. When no *subset* is specified, the state of the system is listed on standard output in three columns. The first column gives the code for a subset known to the system. The second column lists the status of that subset. The value for the status is `installed` if the subset is currently installed, `corrupt` if the subset failed to install correctly, or blank if the subset is not installed. The third column presents the textual description for that subset.

When *subset* arguments are present, the names of the files which make up the contents of each *subset* are listed. The *subset* does not need to be currently installed for this option to display its contents.

–u Update software from media mounted on *location*. Currently installed subsets that are earlier versions than the ones provided on the media will be updated from the distribution.

Update installations replace all files on the system that belong to each subset. Files that are already installed to the system and that have been changed since load time are archived to `/var/adm/install/archive`. All files in the subset to be updated are loaded from the distribution. Any files in the distribution that have been marked by the vendor as having customer precedence are restored to the system from the `/var/adm/install/archive` directory. Any file from the distribution which would be overwritten by restoring an archived file is saved in `/var/adm/install/reference`. All subset inventory information for previously installed copies of the newly updated subsets is lost.

Not all subsets on all distributions are update subsets. If this option is used on a non-update subset, an error message will be printed.

–v Verify each *subset*. The installation verification program (IVP) provided with *subset* is executed. If no IVP is provided for *subset,* `setld` remains silent.

–c Configure *subset,* passing the configuration message *message* to the subset control program.

–x Extract subsets from the distribution media mounted on *location*. If you specify no *subset* arguments, `setld` displays a menu of subsets on the distribution. Subsets chosen from this menu are extracted.

If you specify *subset* arguments, `setld` extracts only the subsets specified.

Description

The `setld` command is used for installing and managing software. Software is organized into *subsets* which may be loaded, deleted, inventoried, updated, verified and configured. The load and update operations read software from disks, tapes or from an Internet installation server.

The `setld` command is also used to read the contents of an installation tape onto a disk so that the disk can be used as the distribution media.

Options

–D *dir* Specify *dir* as the root directory for an operation. The default is / for all operations except **–x**. The default for **–x** is the current directory. If this option is specified, `setld` will operate on the software rooted at the specified directory. This option is useful for installing software to offline systems on removable media or dual-ported disk drives.

Restrictions

No software can be installed or extracted if `setld` would cause the percentage of available space and available gnodes remaining on a file system to fall below 10%.

Do not attempt to install software into an NFS mounted file system.

Examples

Load software subsets from tape unit 2:

```
setld -l /dev/rmt2h
```

Load the `UDTUUCP400` subset from tape unit 2:

```
setld -l /dev/rmt2h UDTUUCP400
```

Load the `UDTUUCP400` subset to an offline system rooted at `/mnt` from tape unit 2:

```
setld -D /mnt -l /dev/rmt2h UDTUUCP400
```

Load the `UDTUUCP400` subset to an offline system rooted at `/mnt` from installation server `mumbly`:

```
setld -D /mnt -l mumbly: UDTUUCP400
```

Load the `UDTUUCP400` subset to an offline system rooted at `/mnt` from a disk distribution in `/mnt2/RISC/BASE`:

```
setld -D /mnt -l /mnt2/RISC/BASE UDTUUCT400
```

Delete the `UDTUUCP400` and `UDTCOMM400` subsets:

```
setld -d UDTUUCP400 UDTCOMM400
```

Delete `UDTUUCP400` and `UDTCOMM400` subsets from the offline system rooted at `/mnt`:

```
setld -D /mnt -d UDTUUCP400 UDTCOMM400
```

Display the status of all subsets known to the system:

```
setld -i
```

Display the status of all subsets known to the offline system rooted at `/mnt`:

```
setld -D /mnt -i
```

Display the contents of the `UDTUUCP400` subset:

```
setld -i UDTUUCP400
```

Update subsets from distribution on tape unit 1:

```
setld -u /dev/rmt1h
```

Update subsets from server `mumbly`:

```
setld -u mumbly:
```

Verify all subsets on the offline system mounted on `/mnt`:

```
setld -D /mnt -v
```

Verify the `ULTVAXC400` subset on the running system:

```
setld -v ULTVAXC400
```

Send the configuration message `Don't Worry, Be Happy` to the `UWSX11400` subset:

```
setld -c UWSX11400 "Don't Worry, Be Happy"
```

Extract subsets from the distribution on tape unit 0 into the current directory:

```
setld -x /dev/nrmt0h
```

Extract subsets from the disk distribution in `/mnt/RISC/UNSUPPORTED` into `/usr/bigdisk`:

```
setld -D /usr/bigdisk -x /mnt/RISC/UNSUPPORTED
```

Return Value

The exit status from `setld` is 0 if the operation requested was performed successfully. All other cases yield exit status of 1 for failed operations on mandatory subsets and greater than 1 for failed operations on optional subsets.

Diagnostics

–c can be used by super-user only
The `setld` command was entered by a non-root user with one of the root-only function keys. The only function which `setld` will perform for non-privileged users is **–i**.

error in Args()
This message is displayed if `setld` cannot understand the command line arguments. It will always be preceded by a usage message or another diagnostic.

Temp directory /usr/tmp/stltmp*XXXXXX* already in use
This message is displayed if the temporary directory that `setld` would create for itself already exists. Run `setld` again.

Cannot create directory *dir*
The directory *dir* which is needed for `setld` to operate correctly could not be created. This can happen if parts of the system are NFS-mounted but not root-mapped.

error in Dirs()
This message always accompanies the preceding two messages.

subset: **not currently installed, cannot configure.**
Occurs when *subset* is used as an argument to **–c** but *subset* is not installed to the system.

subset: **missing control program, cannot configure.**
An attempt is being made to configure *subset* but the program responsible for doing this is missing. Delete the subset and install it again before retrying the operation.

subset: **not currently installed, cannot delete**
A *subset* specified as an argument with the **–d** switch is not installed on the system, it cannot be deleted.

ReadCtrlFile(): cannot find *filename*.ctrl
The control file named in the error message is not where it is expected to be.

ReadCtrlFile(): cannot read *filename*.ctrl
The control file named in the error message exists but cannot be read.

setld: Sorry, You may not delete the *description* (*subset*) subset
The *subset* in the error message which was used as an argument with the **–d** switch is a subset which was marked by its vendor during manufacture as a subset that cannot be deleted. In this case, the subset cannot be removed from the system by `setld`.

subset: **deletion declined by subset control program**
The subset named in the error message cannot be deleted because of the return status of the subset control program. This indicates that the subset control program provided with this subset has determined that the subset should not be deleted. This message may be seen with a diagnostic issued directly from the subset control program. Consult the documentation accompanying the product.

Tape Positioning Error
An error was detected while positioning the tape for a read operation while using `setld` with either the **–l**, **–u**, or **–x** switches. This can indicate a faulty tape or a transient tape subsystem error. Check the error log and try the operation again.

Error Extracting *subset*
An unrecoverable error has occurred when trying to extract a subset from the distribution.

Control Info Error on *subset*
An attempt to access a control or inventory file or subset control program has failed while extracting subsets. This may indicate a faulty distribution. Try the operation again.

subset: **extract checksum error**
A checksum error was detected in the extracted copy of the *subset* subset. This may indicate a transient tape subsystem error. Check the error log and retry the extract.

Error contacting server *hostname*: *error-message*
Attempt to contact installation server *hostname* failed. The *error-message* provides more information.

Device *location* **not supported for installations.**
The *location* specified on the command line was not recognized as a valid input location for a **–l**, **–u**, or **-x** operation.

Cannot access /dev/nrmt*?*h
The device special file `/dev/nrmt`*?* either does not exist or is not a character special file. Remake the file with `MAKEDEV` and try the operation again.

subset: **Unknown subset**
A *subset* argument that was specified with the **–i** switch does not correspond to any subset known to the system. Check the command line for spelling errors.

location/**instctrl: no such file or directory**
The disk distribution *location* specified on the command line does not point to a valid directory. Check the command line for spelling errors.

Files

`/etc/setldlog`	Logfile for `setld` transactions
`/usr/etc/*.inv`	Subset inventory files
`/usr/etc/*.ctrl`	Subset control files
`/usr/etc/*.scp`	Subset control programs
`/usr/etc/*.lk`	Subset installed lock files
`/usr/etc/*.dw`	Subset corrupt lock files
`/var/adm/install/archive`	Update archive directory
`/var/adm/install/reference`	Update reference directory

See Also

kits(1), fitset(8), frm(8), fverify(8), sysupd(8).
Guide to Preparing Software for Distribution on ULTRIX Systems

Name

shmx – shared memory exerciser

Syntax

/usr/field/shmx [**–h**] [**–o***file*] [**–t***i*] [**–m***j*] [**–s***k*] [**–v**]

Description

The `shmx` memory exerciser spawns a background process `shmxb` and these two processes exercise the shared memory segments. They each take turns writing and reading the other's data in the segments.

You can specify the number of memory segments to test and the size of the segment to be tested by `shmx` and `shmxb` processes. The `shmx` exerciser runs until the process receives a <CTRL/C> or a kill -15 *pid.*

A logfile is made in `/usr/field` for you to examine and then remove. If there are errors in the logfile, check the `/usr/adm/syserr/syserr.<hostname>` file, where the driver and kernel error messages are saved. The `shmx` exerciser is automatically invoked when the `memx` exerciser is started. You can also run `shmx` by itself.

Options

–h	Print the help message for the `shmx` command.
–v	Use the `fork`(2) system call instead of `vfork`(2) to spawn `shmxb`.
–o*file*	Save diagnostic output in *file.*
–t*i*	Run time in minutes (*i*). The default is to run until the process receives a <CTRL/C> or a kill -15 *pid.*
–m*j*	The memory segment size in bytes (*j*) to be tested by the processes. Must be greater than 0. The default is SMMAX/6. (SMMAX is a system parameter set in the file `sys/conf/param.c`.)
–s*k*	The number of memory segments (*k*). The default is 6. The maximum is also 6.

Examples

The following example tests six memory segments (default), each with a segment size of SMMAX/6, until a <CTRL/C> or kill -15 *pid* is received:

```
% /usr/field/shmx
```

The following example runs three memory segments of size 100,000 bytes for 180 minutes in the background:

```
% /usr/field/shmx -t180 -m100000 -s3 &
```

Restrictions

If there is a need to run a system exerciser over an NFS link or on a diskless system there are some restrictions. For exercisers that need to write into a file system, such as `fsx`(8), the target file system must be writable by root. Also the directory, in which any of the exercisers are executed, must be writable by root because temporary files are written into the current directory. These latter restrictions are sometimes difficult to overcome because often NFS file systems are mounted in a way that prevents root from writing into them. Some of the restrictions may be overcome by copying the exerciser to another directory and then executing it.

See Also

Guide to System Exercisers

showmount(8nfs)

Name

showmount – show remotely-mounted file systems

Syntax

/usr/etc/showmount [**–a**] [**–d**] [**–e**] [*host*]

Description

The `showmount` command lists all of the NFS client machines that have remotely mounted a filesystem from the NFS server *host* . This information is maintained by the `mountd`(8nfs) daemon on *host,* and is saved across crashes in the `/etc/rmtab` file. The default value for *host* is the value returned by `hostname`(1) or `gethostname`(2).

Options

–d List directories that have been remotely mounted by clients.

–a Print all remote mounts in the format:

```
hostname:directory
```

In this format, *hostname* is the name of the client, and *directory* is the directory that has been mounted.

–e Print the list of file systems exported by the NFS server, *host.*

Restrictions

If an NFS client crashes, `showmount` will incorrectly report that the client still has a file system mounted, until the crashed client reboots.

See Also

hostname(1), exports(5nfs), rmtab(5nfs), mountd(8nfs)

Name

shutdown – close down the system at a given time

Syntax

/etc/shutdown [**-k**] [**-r**] [**-h**] [**-o**] *time [warning-message ...]*

Description

The `shutdown` command provides an automated shutdown procedure that a superuser can use to notify users when the system is shutting down.

The *time* is the time at which `shutdown` will bring the system down. It may be the word 'now', indicating an immediate shutdown, or specify a future time in one of two formats: + *number* or *hour : min.* The first form brings the system down in *number* minutes. The second brings the system down at the time of day indicated, using a 24–hour clock format.

At intervals which get shorter as shutdown nears, warning messages are displayed at the terminals of all users on the system. Warning messages are also sent to users who are logged in to a remote system that has mounted a file system or directory from the local system using NFS. Five minutes before shutdown, or immediately if shutdown is timed for less than five minutes, logins are disabled by creating `/etc/nologin` and writing a message there. If this file exists when a user attempts to log in, `login`(1) prints its contents and exits. The file is removed just before `shutdown` exits.

At shutdown time, a message is written in the file `/usr/adm/shutdownlog`. This message contains the time of shutdown, who ran shutdown, and the reason. Then, a terminate signal is sent at `init` to bring the system to single-user state.

If the `-r`, `-h`, or `-k` options are used, then `shutdown` executes `reboot`(8), `halt`(8), or avoids shutting the system down (respectively). The `-o` option is for use by `opser` only. It indicates to `shutdown` that it is being called by `opser` and not to return to the user.

You should place the time of the shutdown and the warning message in `/etc/nologin`. Use the message to inform the users about when the system will be back up and why it is going down.

Restrictions

You can kill the system only between now and 23:59, if you use the absolute time for shutdown.

Files

`/etc/nologin` Tells login not to let anyone log in

`/usr/adm/shutdownlog`
Log file for successful shutdowns

See Also

login(1), wall(1), halt(8), opser(8), reboot(8), rwalld(8c)

Name

sizer – a program that sizes system hardware

Syntax

/etc/sizer [*options*]

Description

The `sizer` program reports various items of information about a kernel file and the running system. It is also used to create a system configuration file in the doconfig command.

Options

-b Create a boot command file for the current system disk. The `sizer` program creates the appropriate `defboo` and `askboo` command files in `/usr/sys/*.cons.` These files can then be used to update the systems console media. This only applies for VAX 8600, VAX 780, VAX 6200, and VAX 3400 cpus. This also applies for the VAX 750 and VAX 8200 only if the system disk is connected to an HSC.

-c Return the CPU type of the running cpu.

-t *timezone*
Use *timezone* in the config file.

-wt Returns the workstation display type.

-wu Returns the workstation display units.

-k *image*
Use the indicated kernel file instead of `/vmunix.` This is useful when the running system is not `/vmunix` or you wish information about a system that is not running.

-n *filename*
Creates a configuration file. The system must be running the `/vmunix` kernel unless the `-k` option is used. The `-n` option creates a configuration file in `/tmp/filename` and also a file required for `MAKEDEV` called `/tmp/filename.devs.`

NOTE: The user should run `doconfig` to build a new kernel.

-r Return only the name of the root device and partition.

-s Return only the CPU subtype of the running cpu. This only applies to VAX cpus. It returns following values for the following cpus:

Returned Value	CPU
11	VAX8800
10	VAX8700
9	VAX8550
8	VAX8500
6	VAX8300

5	VAX8200
4	VAXstation/MVAX 2000/3100
3	VAX60
1	MVAXII, VAX3x00, VAX62xx

The value 0 is returned for all other cpu types.

Files

```
/usr/sys/780cons/defboo.cmd
/usr/sys/8200cons/defboo.com
/usr/sys/8600cons/defboo.com
/usr/sys/780cons/askboo.cmd
/usr/sys/8200cons/askboo.com
/usr/sys/8600cons/askboo.com
```

See Also

config(8), doconfig(8)
ULTRIX Advanced Installation Guide

Name

snapcopy – copy VAX 8600/8650 snapshot files

Syntax

/etc/snapcopy [**–d**] *directory*

Description

When the system crashes on a VAX 8600 or VAX 8650 system, the console subsystem creates snapshot files containing binary information regarding the state of the hardware at the time of the crash. Snapshot files are typically used by hardware maintenance personnel to analyze and repair the cause of the system crash.

The `snapcopy` command copies any valid snapshot files (called `snap1.dat` and `snap2.dat`) from the console RL02 disk into the directory specified. The files are renamed to show the time of the copy:

hr:min:sec-`snap1.dat`
hr:min:sec-`snap2.dat`

You should delete the comment character for the `snapcopy` command in the `/etc/rc.local` file only for VAX 8600 and VAX 8650 systems.

The *directory* is the name of the directory to which the snapshot files are to be copied.

Options

–d Invalidates the files on the console RL02 disk, meaning that they can be rewritten by the console subsystem in the event of another system crash.

Diagnostics

snapcopy: not a directory
snapcopy: no write permission for directory
snapcopy: could not chdir to directory
The *directory* must exist and be writable.

Files

`/dev/ttyc3`	Special file by which `snapcopy` communicates with the system console terminal
`/dev/crl`	Special file by which `snapcopy` communicates with the console RL02 disk
`/etc/rc.local`	File that contains site-specific commands to be executed when the ULTRIX system is brought to multi-user mode

snmpd(8n)

Name

snmpd – Simple Network Management Protocol (SNMP) Agent for ULTRIX gateways and hosts

Syntax

/etc/snmpd [**–d** *debuglevel logfile*]

Description

The SNMP Agent, `snmpd`, performs SNMP operations on an ULTRIX gateway or host. The daemon, which is started up by an entry in the `/etc/services` file, sits in the background and listens on SNMP port 161. When the `snmpd` daemon receives an SNMP packet from a Network Management Station (NMS), the daemon performs SNMP operations on the packet and returns a valid response to the NMS.

The `snmpd` daemon extracts much of its information from kernel memory. Static variables whose values are not available in the kernel take values from the SNMP configuration file, `/etc/snmpd.conf`.

SNMP Trap Support

The cold start and authentication failure trap types are supported by `snmpd`.

The cold start trap type is generated by `snmpd` when `snmpd` is restarted. The authentication failure trap type is generated when an attempt at using a community fails. The attempt fails when an unauthorized client tries to use `snmpd` or the community is used in a way that the community type does not allow.

The `snmpd` daemon sends traps to all communities specified in the configuration file with a community type `traps`.

The default is for the `snmpd` daemon to generate authentication failure traps. However, if the following clause is specified somewhere in the `/etc/snmpd.conf` file, authentication failure traps are not generated:

```
no_authen_traps
```

SNMP Sets

When the `snmpd` daemon receives a set-request packet, it processes the variables in the packet and verifies that they are valid read-write variables. While performing this verification, the `snmpd` daemon constructs a linked list of the set requests. After it has completed the verification, it performs the actual set operations on the variables, as if they were being performed simultaneously. If any actual set operation fails, all of the previous set variables from the set-request packet are restored to their old values.

SNMP Supported Variables

For a complete listing of the SNMP Management Information Base (MIB) variables that are supported, see the *Guide to Networking*.

Options

By default, the `snmpd` daemon uses the `syslog` command to record its error messages. However, you can obtain certain debugging and trace information by specifying the `-d` flag, the appropriate debug level, and a log file for the output on the `snmpd` command line.

–d *debuglevel logfile*
Outputs debugging and trace information.

You can specify any one of the following debug levels with the `-d` flag:

1 Print the version number, start time, and exit time of `snmpd`. Also print out when an SNMP packet is received, the address of the sender, and the packet size in bytes.

2 Print out what the `snmpd` daemon has read from the `/etc/snmp.conf` file.

3 Dump the SNMP packet that the `snmpd` daemon has just received and is about to process. Also print out the route and interface address that the `snmpd` daemon is currently looking up. This debug level also dumps the SNMP packet that the server is sending back in response to a received SNMP message.

4 Dump the `snmp` variable tree. Also print out the static bootstrap array of tree information.

The output for a debug level includes the information for all levels including and below the level that you specify. For example, if you specify a debug level of 3, your output includes debug information for levels 3, 2, and 1.

If no debug levels are set, `snmpd` detaches itself from the controlling terminal and executes in the background.

Restrictions

Not all of the MIB variables are supported.

Only the `_mgmt_mib_interfaces_ifTable_ifEntry_ifAdminStatus` variable is settable.

Files

`/etc/snmpd.conf`	SNMP configuration file

See Also

snmpext(3n), snmpd.conf(5n), snmpsetup(8n)
RFC 1066—*Management Information Base for Network Management of TCP/IP-based Internets*
RFC 1067—*A Simple Network Management Protocol*
Guide to Networking

snmpsetup (8n)

Name

snmpsetup – set up the Simple Network Management Protocol (SNMP) Agent

Syntax

/usr/etc/snmpsetup

Description

The `snmpsetup` command is an interactive facility for configuring the ULTRIX SNMP Agent. When your system is configured as an SNMP Agent certain network parameters can be monitored or managed by a Network Management Station (NMS).

The `snmpsetup` command also allows you to configure Extended ULTRIX SNMP Agents. Extended Agents are user-defined daemons that allow you to specify private network parameters to be monitored or managed by the NMS.

Before you invoke `snmpsetup`, the network must be up and running, your host must have a name, and you must know the Internet Protocol address of any NMSs that you want to have monitor or manage your system.

You can run `snmpsetup` while in multiuser mode, but you must be root or superuser.

To invoke `snmpsetup`, type:

```
# snmpsetup
```

Diagnostics

The following error messages are issued by `snmpsetup`:

snmpsetup must be run by root
snmpsetup: need /etc/rc
snmpsetup: need /etc/snmpd.conf
snmpsetup: non-interactive mode not currently supported
snmpsetup: network must be setup before running snmpsetup
snmpsetup: terminated with no installations made

Files

`/etc/snmpd.conf`	SNMP configuration file
`/etc/rc`	File that controls automatic reboot

See Also

snmpext(3n), snmpd.conf(5n), snmpd(8n)
Guide to Networking

Name

startcpu – start one or all CPUs

Syntax

/etc/**startcpu** [*cpunumber*]

Description

The `startcpu` program is used to start the non-boot CPUs in a system. Normally, it will be executed from the `/etc/rc` startup file as the system comes up in multi-user mode.

If no argument is used, `startcpu` will try to start all the non-boot CPUs in the system. Otherwise, it starts the specified CPU.

See Also

startcpu(2), stopcpu(2), stopcpu(8)

statd(8c)

Name

statd – network status monitor daemon

Syntax

/usr/etc/statd

Description

The `statd` daemon monitors the status of the client and server sites in response to a request made by the local `lockd` daemon. When a site failure is detected, `statd` notifies the local `lockd` daemon, which then processes the recovery of the locked files or file regions.

Restrictions

The crash of a site is only detected on its recovery.

Files

```
/etc/sm
/etc/sm.back
/etc/state
```

See Also

statmon(5), lockd(8C)

Name

sticky – executable files with persistent text

Description

The sticky bit (file mode bit 01000), is used to indicate special treatment for certain executable files and directories.

While the sticky bit, mode 01000 is set on a sharable executable file, the text of that file will not be removed from the system swap area. Thus the file does not have to be fetched from the file system upon each execution. As long as a copy remains in the swap area, the original text cannot be overwritten in the file system, nor can the file be deleted. Directory entries can be removed so long as one link remains.

Sharable files are made by the `-n` and `-z` options of `ld`(1).

To replace a sticky file that has been used, clear the sticky bit with `chmod` and execute the old program to flush the swapped copy. This can be done safely even if others are using it. Overwrite the sticky file. If the file is being executed by any process, writing will be prevented. It suffices to simply remove the file and then rewrite it, being careful to reset the owner and mode with `chmod` and `chown`. Set the sticky bit again.

A directory whose sticky bit is set becomes an append-only directory, or, more accurately, a directory in which the deletion of files is restricted. A file in a sticky directory may only be removed or renamed by a user if the user has write permission for the directory and the user is the owner of the file, the owner of the directory, or the superuser. This feature is usefully applied to directories such as `/tmp` which must be publicly writeable but should deny users the license to arbitrarily delete or rename each others' files.

Restrictions

Only the superuser can set the sticky bit.

See Also

chmod(2)

sticky(8)

Name

sticky – executable files with persistent text

Description

The sticky bit (file mode bit 01000), is used to indicate special treatment for certain executable files and directories.

While the sticky bit, mode 01000 is set on a sharable executable file, the text of that file will not be removed from the system swap area. Thus the file does not have to be fetched from the file system upon each execution. As long as a copy remains in the swap area, the original text cannot be overwritten in the file system, nor can the file be deleted. Directory entries can be removed so long as one link remains.

Sharable files are made by the **–n** and **–z** options of `ld`(1).

To replace a sticky file that has been used, clear the sticky bit with `chmod` and execute the old program to flush the swapped copy. This can be done safely even if others are using it. Overwrite the sticky file. If the file is being executed by any process, writing will be prevented. It suffices to simply remove the file and then rewrite it, being careful to reset the owner and mode with `chmod` and `chown`. Set the sticky bit again.

A directory whose sticky bit is set becomes an append-only directory, or, more accurately, a directory in which the deletion of files is restricted. A file in a sticky directory may only be removed or renamed by a user if the user has write permission for the directory and the user is the owner of the file, the owner of the directory, or the superuser. This feature is usefully applied to directories such as `/tmp` which must be publicly writable but should deny users the license to arbitrarily delete or rename each others' files.

Restrictions

Only the superuser can set the sticky bit.

Is largely unnecessary on the VAX. It matters only for large programs that will page heavily to start, since text pages are normally cached incore as long as possible after all instances of a text image exit.

See Also

chmod(2)

Name

stopcpu – stop one or all CPUs

Syntax

/etc/stopcpu [*cpunumber*]

Description

The `stopcpu` program is used to stop the non-boot CPUs in a system. If no argument is used, `stopcpu` will try to stop all the non-boot CPUs in the system. Otherwise, it stops the specified CPU.

See Also

startcpu (2), stopcpu (2), startcpu (8)

Name

svcsetup – set up the svc.conf file

Syntax

/usr/etc/svcsetup [[–d *directory*] **–o** *name_service_order*]

Description

The `svcsetup` command allows you to print and modify the contents of the `svc.conf` file on the current system. This file must be modified when adding or removing a naming service, such as Yellow Pages or BIND/Hesiod. The security parameters also included in the `svc.conf` file can only be changed by `secsetup`. Changes take effect immediately.

Modifications to this file can also be made via an editor.

You can supply the name service ordering for all databases from the command line using the `-o` option. The combination of the `-o` and `-d` options allows you to set up a diskless client from the diskless server.

When you run `svcsetup` interactively, and choose `modify` from the configuration menu, you can choose to modify any number of the databases listed by entering each number separated by space at the prompt. When choosing a naming service order for a database, enter one number from the menu choices shown which corresponds to the naming service order desired.

Options

–d *directory*
These two arguments are required if you are setting up a diskless client from the diskless server. The *directory* is the full pathname of the root directory for your system (a diskless client) on the diskless server. The following is an example of a root directory for a diskless client named `orange`:

```
/dlclient0/orange.root
```

–o *name_service_order*
This is the name service order to be set for all database entries in the `svc.conf` file. The name service order can be one of the following, which are the only valid strings for *name_service_order* argument:

- local
- local,yp
- local,bind

Restrictions

The recommended configuration is that you have `local` as the first entry for all databases.

You must have `local` as the first entry for the `passwd` and `hosts` databases.

You must have `yp` as the entry for the `netgroup` database.

You must have either `local` or `bind` as the entry for the `auth` database.

You must be superuser to run `svcsetup`.

Files

`/etc/svc.conf`	Services order configuration file
`/usr/sys/h/svcinfo.h`	System header file

See Also

svc.conf(5)

swapon(8)

Name

swapon – specify additional device for paging and swapping

Syntax

/etc/swapon –a
/etc/swapon *name* ...

Description

The `swapon` command is used to specify additional devices on which paging and swapping are to take place. The system begins by swapping and paging on only a single device so that only one disk is required at bootstrap time. Calls to `swapon` normally occur in the system multi-user initialization file `/etc/rc`, making all swap devices available, so that the paging and swapping activity is interleaved across several devices.

Normally, the `-a` option is given, causing all devices marked as ''sw'' (swap devices) in `/etc/fstab` to be made available.

The second form gives individual block devices, as listed in the system swap configuration table. The call makes only this space available to the system for swap allocation.

Restrictions

There is no way to stop paging and swapping on a device. It is therefore not possible to make use of devices which may be dismounted during system operation.

Files

`/dev/[ru][pk]?b`	Normal paging devices

See Also

swapon(2), init(8)

Name

syscript – dialogue for running system exercisers

Syntax

syscript

Description

The `syscript` command presents a dialogue that lets you run any of the following system exercisers:

cmx(8)	Tests the terminal communications system
dskx(8)	Tests the disk drives
fsx(8)	Tests the file systems
lpx(8)	Tests the line printers
memx(8)	Tests memory
mtx(8)	Tests the magnetic tape drives
netx(8)	Tests the TCP/IP network system

To execute the `syscript` command, you must be logged in as the superuser, and your working directory must be the `/usr/field` directory.

See Also

Guide to System Exercisers

Name

syslog – log systems messages

Syntax

/etc/syslog [**–m***N*] [**–f***name*] [**–d**]

Description

The `syslog` command reads a datagram socket and logs each line it reads into a set of files described by the configuration file `/etc/syslog.conf`. The `syslog` command configures when it starts up and whenever it receives a hangup signal.

Each message is one line. A message can contain a priority code, marked by a digit in angle braces at the beginning of the line. Priorities are defined in < syslog.h >, as follows:

LOG_ALERT	This priority should essentially never be used. It applies only to messages that are so important that every user should be aware of them, for example, a serious hardware failure.
LOG_SALERT	Messages of this priority should be issued only when immediate attention is needed by a qualified system person, for example, when some valuable system resource disappears. These messages are sent to a list of system people.
LOG_EMERG	Emergency messages are not sent to users, but represent major conditions. An example might be hard disk failures. These could be logged in a separate file so that critical conditions could be easily scanned.
LOG_ERR	These messages represent error conditions, such as soft disk failures, etc.
LOG_CRIT	Such messages contain critical information, but which can not be classed as errors, for example, 'su' attempts. Messages of this priority and higher are typically logged on the system console.
LOG_WARNING	These messages are issued when an abnormal condition has been detected, but recovery can take place.
LOG_NOTICE	These messages fall into the class of ''important information''; this class is informational but important enough that you don't want to throw it away casually. Messages without any priority assigned to them are typically mapped into this priority.
LOG_INFO	These are information level messages. These messages could be thrown away without problems, but should be included if you want to keep a close watch on your system.
LOG_DEBUG	These messages may be useful to log certain debugging information. Normally this information is thrown away.

It is expected that the kernel will not log anything below LOG_ERR priority.

The configuration file is in two sections separated by a blank line. The first section defines files that `syslog` will log into. Each line contains a single digit which defines the lowest priority (highest numbered priority) that this file will receive, an optional asterisk which guarantees that something gets output at least every 20 minutes, and a pathname. The second part of the file contains a list of users that will be informed on SALERT level messages. For example, the following logs all messages of priority 5 or higher onto the system console, including timing marks every 20 minutes:

```
5*/dev/console
8/usr/spool/adm/syslog
3/usr/adm/critical

eric
kridle
kalash
```

This example logs all messages of priority 8 or higher into the file `/usr/spool/adm/syslog;` and all messages of priority 3 or higher into `/usr/adm/critical.` The users "eric", "kridle", and "kalash" will be informed on any subalert messages.

The flags are:

–m Set the mark interval to *N* (default 20 minutes).

–f Specify an alternate configuration file.

–d Turn on debugging (if compiled in).

To bring `syslog` down, it should be sent a terminate signal. It logs that it is going down and then waits approximately 30 seconds for any additional messages to come in.

There are some special messages that cause control functions. "<*>N" sets the default message priority to *N*. "<$>" causes `syslog` to reconfigure (equivalent to a hangup signal). This can be used in a shell file run automatically early in the morning to truncate the log.

The `syslog` command creates the file `/etc/syslog.pid` if possible containing a single line with its process ID. This can be used to kill or reconfigure `syslog`.

Restrictions

LOG_ALERT and LOG_SUBALERT messages should only be allowed to privileged programs.

Actually, `syslog` can not deal with kernel error messages in the current implementation.

Files

`/etc/syslog.conf`
Configuration file

`/etc/syslog.pid`
Process id

See Also

syslog(3)

Name

talkd – inter-terminal communications server

Syntax

/etc/talkd

Description

The `talkd` program is the server for the `talk(1)` program. The server provides a rendezvous method for the requesting (possibly remote) `talk` and the local responding `talk`.

The `talkd` server is invoked by `inetd`(8c) when it receives a packet on the port indicated in the talk service specification.

Restrictions

The `talkd` server does not strictly follow network byte order in its packet format and may have difficulty in talking with implementations of `talkd` on other architectures that do not take this into account.

The version of `talk` released with ULTRIX V3.0 uses a protocol that is incompatible with the protocol used in earlier versions. Starting with ULTRIX V3.0, the `talk` program communicates with other machines running ULTRIX, V3.0 (and later), and machines running 4.3 BSD or versions of UNIX based on 4.3 BSD.

The `talk` command is not 8-bit clean. Typing in DEC Multinational Characters (DECMCS) causes the characters to echo as a sequence of a carets (^) followed by the character represented with its high bit cleared. This limitation makes `talk` unusable if you want to communicate using a language which has DECMCS characters in its alphabet.

See Also

talk(1), services(5), inetd(8c)

Name

tapex – tape exerciser program

Syntax

tapex [*option(s)*] [*parameter(s)*]

Description

The `tapex` program tests tape driver functionality. These tests provide more comprehensive functional coverage than the `mtx` utility which does simple start/stop oriented read/write testing. Functions that are tested include:

- Writing records onto a tape and verifying the records
- Using records in a range of sizes
- Record-length testing
- Random record-size testing
- Positioning tests for records and files
- Writing and reading past the end of media
- End-of-file testing
- n-buffered I/O testing
- Tape-transportability testing
- Bandwidth performance analysis
- Media loader testing
- Reporting of tape contents

When `tapex` is run, a writable tape must be loaded in the drive being tested, and the drive must be online.

Options

Some `tapex` options cause specific tests to be performed, for example, an end-of-media test. Other options modify the tests, for example, enabling caching. The `tapex` options are as follows:

–a Performance measurement test that calculates the tape transfer bandwidth for writes and reads to the tape by timing data transfers.

–b Continuously runs the write/read tests until the process is killed. This flag can be used in conjunction with the `-r` or `-g` flag.

–c Enables caching on the device, where supported. This does not specifically test caching, but it enables the use of caching on a tape device while running the other tests.

–C Disables caching on TMSCP tape devices. If the tape device is a TMSCP unit, then caching is the default mode of test operation. This flag causes the tests to be run in noncaching mode.

–d Tests the ability to append to the media. First, the test writes records to the tape. Then, it repositions back one record and appends additional records. Finally, the test does a read verification. This test simulates the behavior of the `tar r` switch.

–e End-of-media test. This test first writes data to fill up a tape, which may take a long time for long tapes. It then does reads and writes past the end of media, which should fail. Next it enables writing past end of media, writes to the tape, and reads back the records for validation.

–E Runs an extensive series of tests in sequential order. Due to the large number of tests, this option takes a long time to complete. Depending on tape type and cpu type, this series of tests can take up to 10 hours to complete.

–f `/dev/rmt#`*?FP*
Specifies the name of the device special file that corresponds to the tape unit being tested. The number sign (#) symbol represents the unit number. The question mark (?) argument can be the letter `h` for the high density device or the letter `l` for the low density device. The default tape device is `/dev/rmt0h`.

–F File-positioning tests. First, files are written to the tape and verified. Next, every other file on the tape is read. Then, the previously unread files are read by traversing the tape backwards. Finally, random numbers are generated; the tape is positioned to those locations, and the data is verified. Each file uses a different record size.

–G File-positioning tests on already-written tape. This flag can be used in conjunction with the `-F` flag to run the file position tests on a tape that has already been written to by a previous version of the `-F` test. For this to work, the same test parameters, for example record size and number of files, must be used as when the the tape was written. No other data should have been written to the tape since the previous `-F` test.

–g Random record-size tests. This test writes records of random sizes. It reads in the tape, specifying a large read size; however, only the amount of data in the randomly-sized record should be returned. This test only checks return values and does not validate record contents.

–h Displays a help message describing the tape exerciser.

–i Interactive mode. Under this mode, the user is prompted for various test parameters. Typical parameters include the record size and the number of records to write. The following scaling factors are allowed:

k or **K**	for kilobyte (1024 * n)
b or **B**	for block (512 * n)
m or **M**	for megabyte (1024 * 1024 * n)

For example, 10k would specify 10240 bytes.

–j Write phase of the tape-transportability tests. This test writes a number of files to the tape, and then verifies the tape. After a successful verification, the tape is brought offline to be moved to another tape unit and read in with the `-k` option. The purpose of this test is to prove that a tape can be written on one drive and read by another drive. Note that the test parameters for the `-k` phase of the transportability test must match the parameters of the `-j` test. Any changes of test parameters from the defaults should also be changed during the `-k` test.

–k Read phase of the tape-transportability tests. This test reads a tape that was written by the `-j` test and verifies that the expected data is read from the tape. Success of this test proves that a tape can be written on one drive and read on another. As stated in the the description of the `-j` option, any parameters changed in the `-j` test must also be changed in the `-k` test.

–L Media loader test. For sequential stack loaders, the media is loaded, written to, and verified. Then the media is unloaded, and the test repeats on the next piece of media. This verifies that all the media in the input deck is writable. To run this test in read-only mode, also specify the `-w` option.

–l End-of-file test. This test verifies that a zero byte count is returned when a tape mark is read and that another read will fetch the first record of the next tape file.

–m Displays tape contents. This is not a test; it reads the tape sequentially and prints out the number of files on the tape, the number of records in each file, and the size of the records within the file. The contents of the tape records are not examined.

–N Disables the usage of n-buffered I/O on tests that support its usage. (See `nbuf`(4) for a description of n-buffered I/O.)

–o *filename*
Sends output to the specified filename. The default is to not create an output file and send output to the terminal.

–p Runs both the record and file positioning tests. (See the `-R` and `-F` options.)

–q Command timeout test. This test verifies that the driver allows enough time for completion of long operations. The test consists of writing files to fill up the tape. Next a rewind is performed followed by a forward skip out to the last file. The test is successful if the forward skip operation completes without error.

–r Record-size test. A number of records are written to the tape and then verified. This process is repeated over a range of record sizes.

–R Record-positioning test. First, records are written to the tape and verified. Next, every other record on the tape is read. Then, the other records are read by traversing the tape backwards. Finally, random numbers are generated; the tape is positioned to those locations, and the data is verified.

–s Record-size behavior test. Verifies that a record read will return at most one record or the read size, whichever is less.

–S Single record size test. This option modifies the record-size test (`-r` option).

–T Copies output to standard output. This flag is useful if you want to log output to a file with the `-o` option and also have the output displayed on standard output. This flag must be specified after the `-o` flag in the command line.

–v Verbose mode. This option causes more detailed terminal output of what the tape exerciser is doing. For example, it lists operations the exerciser is performing, such as record counts, and more detailed error information.

–V Very verbose mode. This option causes more output to be generated than either the default mode or the `-v` flag. The output consists of additional status information on exerciser operation.

–w	Opens the tape as read-only. This mode is only useful for tests that do not write to the media. For example, it allows the `-m` test to be run on a write-protected media.
–Z	Initializes read buffer to the nonzero value 0130. This may be useful for debugging purposes. If the `-Z` flag is not specified, all elements of the read buffer will be initialized to 0. Many of the tests first initialize their read buffer and then perform the read operation. After reading a record from the tape, some tests validate that the unused portions of the read buffer remain at the value to which they were initialized. As a debugging tool, it may in some cases be useful to have this initialized value set to be nonzero. In those cases, the arbitrary character 0130 can be used.

Parameters

You can change the default test parameters either by using the `-i` option described previously or by specifying the parameters in the command line. This section describes the parameters you can set in the command line, listed with the associated test.

To specify a value, type the parameter name followed by a space and then the number. For example `-min_rs` 512 specifies a minimum record size of 512 blocks. The following scaling factors are allowed:

k or **K**	for kilobyte (1024 * n)
b or **B**	for block (512 * n)
m or **M**	for megabyte (1024 * 1024 * n)

For example, 10K would specify 10240 bytes.

These parameters are associated with the option `-a`:

–perf_num	The number of records to write and read.
–perf_rs	The size of records.

These parameters are associated with the option `-d`:

–tar_num	The number of additional and appended records.
–tar_size	The record size for all records written in this test.

These parameters are associated with the option `-e`.

Note that specifying too much data to be written past EOM could cause a reel-to-reel tape to go off the end.

–end_num	The number or records written past EOM.
–end_rs	The record size.

These parameters are associated with the option `-F`:

–num_fi	The number of files.
–pos_ra	The number of random repositions.
–pos_rs	The record size.
–rec_fi	The number of records per file.

This parameter is associated with the option `-g`:

–rand_num	The number of records to write and read.

These parameters are associated with the options `-j` and `-k`:

–tran_file	The number of files to write or read.
–tran_rec	The number of records contained in each file.
–tran_rs	The size of each record.

These parameters are associated with the option `-R`:

–pos_num	The number of records.
–pos_ra	The number of random repositions.
–pos_rs	The record size.

These parameters are associated with the options `-r` and `-S`:

–inc	The record increment factor.
–max_rs	The maximum record size.
–min_rs	The minimum record size.
–num_rec	The number of records.
–t	Sets a time limit in minutes on how long to run the record-size test (`-r` option). The default is to run the test to completion.

These parameters are associated with the option `-s`:

–num_rec	The number of records.
–size_rec	The record size.

This parameter is used in any test which supports n-buffered I/O:

–num_nbuf	The number of buffers to use.

This parameter is associated with all tests:

–err_lines	The threshold on error printouts.

Examples

This example runs a series of tests on tape device `rmt1h` and sends all output to a file called `tapex.out`.

```
tapex -f /dev/rmt1h -E -o tapex.out
```

This example runs the end-of-media test on tape device `rmt4h`. Verbose mode is specified, which causes additional output. By default, output is directed to the terminal.

```
tapex -f /dev/rmt4h -v -e
```

This example performs read/write record testing. By default, this test runs on the default tape device `/dev/rmt0h` and output is sent to the terminal.

```
tapex -r
```

This example performs read/write record testing using record sizes in the range 10k to a maximum record size of 20k. By default, this test runs on the default tape device `/dev/rmt0h` and output is sent to the terminal.

```
tapex -r -min_rs 10k -max_rs 20k
```

See Also

mtx(8)
Guide to System Exercisers

telnetd (8c)

Name

telnetd – DARPA TELNET protocol server

Syntax

/etc/telnetd

Description

The `telnetd` server supports the DARPA standard TELNET virtual terminal protocol. The TELNET server is invoked when `inetd`(8c) receives a connection request on the port indicated in the TELNET service description.

The `telnetd` server operates by allocating a pseudo-terminal device for a client, then creating a login process which has the slave side of the pseudo-terminal as **stdin**, **stdout**, and **stderr**. The `telnetd` server manipulates the master side of the pseudo terminal, implementing the TELNET protocol and passing characters between the client and login process.

When a TELNET session is started up, `telnetd` sends a TELNET option to the client side indicating a willingness to do remote echo of characters, to suppress go ahead, and to receive terminal type information from the remote client. If the remote client is willing, the remote terminal type is propagated in the environment of the created login process. The pseudo terminal allocated to the client is configured to operate in cooked mode and with XTABS and CRMOD enabled.

Aside from this initial setup, the only mode changes `telnetd` will carry out are those required for echoing characters at the client side of the connection.

The `telnetd` server supports binary mode, suppress go ahead, echo, and timing mark. It also allows a remote client to do binary, terminal type, and suppress go ahead.

Restrictions

Some TELNET commands are only partially implemented.

The TELNET protocol allows the exchange of the number of lines and columns on the user's terminal, but `telnetd` does not make use of them.

The terminal type name received from the remote client is converted to lower case.

The `telnetd` server never sends TELNET go ahead commands.

See Also

telnet(1c), pty(4), tty(4), services(5), inetd(8c)

Name

tftpd – DARPA Trivial File Transfer Protocol (TFTP) server

Syntax

/usr/etc/tftpd [**–r** *pathname*]

Description

The server, `tftpd`, supports the DARPA Trivial File Transfer Protocol (TFTP). The TFTP server is invoked when `inetd`(8c) receives a packet on the port indicated in the TFTP service description. The server is not normally started by default from `inetd`(8c).

The use of `tftp` does not require an account or password on the remote system. Because of the lack of authentication information, `tftpd` will allow only publicly readable files to be accessed. This extends the concept of public to include all users on all hosts that can be reached through the network. This may not be appropriate on all systems however, and its implications should be considered before enabling TFTP service.

The server should have the user ID with the lowest possible privilege.

Options

–r *pathname* The *pathname* can be the choice of the user. For example, `/guests/ftp` would allow only files below `/guests/ftp` to be copied using `tftp`.

Restrictions

This server is known only to be self consistent, that is, it operates with the user TFTP program, `tftp`(1c). Because of the unreliability of the transport protocol (UDP) and the scarcity of TFTP implementations, it is uncertain whether it really works.

The search permissions of the directories leading to the files accessed are not checked.

See Also

tftp(1c), services(5), inetd(8c)

timed (8)

Name

timed – time server daemon

Syntax

/usr/etc/timed [**–i** | **–n** *network*][**–E**][**–M**][**–t**]

Description

The `timed` daemon synchronizes a host's time with the time of other machines in a local area network running `timed`. It is normally invoked at boot time from the `/etc/rc.local` file.

Servers running `timed` slow down the clocks of some machines and speed up the clocks of others to bring them all to the average network time. The average network time is computed from measurements of clock differences with the Internet Communication Message Protocol (ICMP) timestamp request message.

The service provided by `timed` is based on a master-slave scheme. When `timed` is started on a machine, it asks the master for the network time and sets the host's clock to that time. After that, it accepts synchronization messages periodically sent by the master and calls the `adjtime` or `settimeofday` routine to perform any corrections on the host's clock.

It also communicates with the `date` command to set the date globally, and with the `timed` control program, `timedc`. If the machine running the master crashes, then the slaves elect a new master from among slaves running with the `-M` flag set. The `-M` flag provides time synchronization on any attached networks where no current master server is detected. Such a server propagates the time computed by the top-level master. At least one `timed` daemon on each network must run with the `-M` option set to allow it to become a `timed` master.

Options

–E Overrides the input of slaves. Use the `-E` flag in conjunction with the `-M` flag. It specifies that a master `timed` should not average the times of the slaves to calculate the network time but should distribute the time of its local host as the network time. This flag allows a master `timed` to distribute time to a network while the network time is controlled by an outside agent such as the Network Time Protocol.

[–i | –n] *network*

–i Specifies a network to ignore. Each network that appears as an argument to the `-i` flag is added to the list of networks that `timed` will ignore. If the `-i` flag is used, `timed` accesses all networks to which the host is connected except for those networks specified as arguments to the `-i` flag.

–n Specifies a network to use. When the `timed` is started, it gathers information about all the network devices connected to the local host. If neither the `-n` flag nor the `-i` flag is used, `timed` tries to access all the network devices connected to the local host. The network argument to the `-n` flag is the name of a network that `timed` should access.

If the `-n` switch is used, only those networks specified by the `-n` flag are accessed.

Do not use the `-i` and `-n` flags together.

–M Allows a slave time server to become a master time server if the master server crashes. A system running the `timed` daemon without the `-M` flag set remains a slave. The `timed` daemon checks for a master time server on each network to which it is connected. It requests synchronization service from the first master server it locates.

–t Enables `timed` to trace the messages it receives in the file `/usr/adm/timed.log`. Tracing can be enabled or disabled with the `timedc` program.

Restrictions

Any system running `timed` with the `-E` and `-M` options set is eligible to become the `timed` master, and distribute its local time to all systems running `timed` on its network. Run the Network Time Protocol daemon, `ntpd`, instead of `timed` to prevent this behavior.

Files

`/etc/rc.local`	Invokes the `timed` daemon each time the system boots
`/usr/adm/timed.log`	Tracing file for `timed`
`/usr/adm/timed.masterlog`	Log file for master `timed`

See Also

date(1), adjtime(2), gettimeofday(2), settimeofday(2), networks(5), ntpd(8), timedc(8)
Introduction to Networking and Distributed System Services

Name

timedc – timed control program

Syntax

/usr/etc/timedc [*command* [*argument* ...]]

Description

The `timedc` program controls the operation of the `timed` daemon. If you run `timedc` without any arguments, `timedc` enters interactive mode and displays the `timedc>` prompt.

If you supply a `timedc` command on the command line, `timedc` runs the command and then exits. If you redirect the standard input of `timedc` from an interactive terminal to a file, `timedc` interprets the contents of the file as a list of commands separated by carriage returns and terminated with an EOF character.

Commands

?[*command...*]

help[*command...*]
Prints a short description of each command specified in the argument list. If no arguments are given, a list of the recognized commands is printed.

clockdiff[*host...*]
Computes the differences between the clock of the host machine and the clocks of the machines given as arguments.

trace[on | off]
Enables or disables the logging of incoming messages to `timed`. The `trace` command logs messages in the file `/usr/adm/timed.log`.

[incr | decr] [–cd][*minutes:*][*seconds.*][*microseconds*]
Increments or decrements the value of the local clock so that the clock gains or loses the specified amount of time.

–c
Adjusts the local clock continuously. Adjustments specified with the `-c` switch should be on the order of microseconds, and are added or subtracted from the local clock in small stages. This type of adjustment avoids large instantaneous jumps and guarantees that the graph of local clock time versus real time remains continuous.

The following example increments the local clock continuously by 500 microseconds:

```
/etc/timedc incr -c 500
```

–d
Adjusts the local clock instantaneously. Adjustments specified with the `-d` switch should be on the order of seconds, and are added or subtracted from the local clock at once. The graph of local clock time versus real time is discontinuous.

The following example decrements the local clock discontinuously by five minutes and two seconds:

```
/etc/timedc decr -d 5:2
```

msite	Indicates which site the master is running on currently.
quit	Exits from the `timedc` program.

Diagnostics

?Ambiguous command
Abbreviation matches more than one command.

?Invalid command
No match was found.

?Privileged command
Command can be executed by root only.

Files

`/usr/adm/timed.log`	Tracing file for `timed`
`/usr/adm/timed.masterlog`	Log file for master `timed`

See Also

date(1), adjtime(2), settimeofday(2), timed(8)

trigger (8)

Name

trigger – trigger a target node to request a down-line load

Syntax

/etc/trigger *node* [*options*]

Description

The `trigger` command triggers the bootstrap mechanism of a target node, causing the target to request a down-line load. Once a target node is triggered, it loads itself in whatever manner its primary loader is programmed to operate. The target node could request a down-line load from the host that just triggered it or from another adjacent node, or the target node could load itself from its own mass storage device.

The *node* argument is the name or address of the target node. A node name consists of from one to six alphanumeric characters. For single networks, a node address consists of a decimal integer (1-1023). For multiple networks, a node address consist of two decimal integers (n.n), where the first indicates the network number (1-63), and the second indicates the node address (1-1023).

The `trigger` command requires the identification of the service circuit over which the load is performed, the Ethernet hardware address of the target node, and the service password needed to gain access to the target. This information is included in the nodes database entry for the target node. A node entry is defined with the `addnode` command. For further information, see `addnode(8)`. Alternatively, you can choose not to include a target node's service password in the nodes database for security reasons. You must therefore specify this value in the command line by using the **–p** option.

Options

–p Uses the specified service and password (next arguments) in accessing the target node.

Examples

This command triggers node Bangor to issue a down-line load request:

```
# /etc/trigger bangor <RET>
```

See Also

mop_mom(8), addnode(8), load(8), remnode(8), getnode(8), ccr(8)
Guide to Local Area Transport Servers

Name

trpt – transliterate protocol trace

Syntax

trpt [**–a**] [**–s**] [**–t**] [**–j**] [**–p** *hex-address*] [*system* [*core*]]

Description

The `trpt` command interrogates the buffer of TCP trace records created when a socket is marked for debugging and prints a readable description of these records.

Options

When no options are supplied, `trpt` prints all the trace records found in the system grouped according to TCP connection protocol control block (PCB). The following options may be used to alter this behavior:

–s	In addition to the normal output, prints a detailed description of the packet sequencing information.
–t	In addition to the normal output, prints the values for all timers at each point in the trace.
–j	Gives a list of the protocol control block addresses for which there are trace records.
–p	Shows only trace records associated with the protocol control block whose address follows.
–a	In addition to the normal output, prints the values of the source and destination addresses for each packet recorded.

The recommended use of `trpt` is to isolate the problem and enable debugging on the socket(s) involved in the connection. Find the address of the protocol control blocks associated with the sockets using the **–A** option to `netstat(1)`. Then run `trpt` with the **–p** option, supplying the associated protocol control block addresses. If there are many sockets using the debugging option, the **–j** option may be useful in checking to see if any trace records are present for the socket in question.

If debugging is being performed on a system or core file other than the default, the last two arguments may be used to supplant the defaults.

Diagnostics

no namelist
The system image does not contain the proper symbols to find the trace buffer.

Other diagnostic messages are self-explanatory.

Files

```
/vmunix
/dev/kmem
```

See Also

netstat(1), setsockopt(2)

Name

tunefs – tune up an existing file system

Syntax

/etc/tunefs [*options*]

Description

The `tunefs` command is designed to change the dynamic parameters of a file system which affect the layout policies. The parameters which are to be changed are indicated by the options listed in the following section.

Options

–a maxcontig
This specifies the maximum number of contiguous blocks that will be laid out before forcing a rotational delay (see **–d** below). The default value is one, since most device drivers require an interrupt per disk transfer. Device drivers that can chain several buffers together in a single transfer should set this to the maximum chain length.

–d rotdelay
This specifies the expected time (in milliseconds) to service a transfer completion interrupt and initiate a new transfer on the same disk. It is used to decide how much rotational spacing to place between successive blocks in a file.

–e maxbpg This indicates the maximum number of blocks any single file can allocate out of a cylinder group before it is forced to begin allocating blocks from another cylinder group. Typically this value is set to about one quarter of the total blocks in a cylinder group. The intent is to prevent any single file from using up all the blocks in a single cylinder group, thus degrading access times for all files subsequently allocated in that cylinder group. The effect of this limit is to cause big files to do long seeks more frequently than if they were allowed to allocate all the blocks in a cylinder group before seeking elsewhere. For file systems with exclusively large files, this parameter should be set higher.

–m minfree
This value specifies the percentage of space held back from normal users; the minimum free space threshold. The default value used is 10%. This value can be set to zero, however up to a factor of three in throughput will be lost over the performance obtained at a 10% threshold. Note that if the value is raised above the current usage level, users will be unable to allocate files until enough files have been deleted to get under the higher threshold.

–c Clean byte timeout factor. The metrics used to determine if a clean byte associated with a file system should be invalidated, decrement a timeout factor when crossed. When the timeout factor reaches zero, the clean byte is invalidated and `fsck` will automatically check the file system. The timeout factor can be increased to a value between 0 and 255. A value of zero will cause `fsck` to check the file system on every reboot.

Restrictions

This program should work on mounted and active file systems. Because the super-block is not kept in the buffer cache, the program will only take effect if it is run on dismounted file systems. If run on the root file system, the system must be rebooted.

See Also

fs(5), fsck(8), mkfs(8), newfs(8)
''A Fast File System for UNIX,'' *ULTRIX Supplementary Documents, Volume 3: System Manager*

Name

uerf – ULTRIX error report formatter

Syntax

/etc/uerf [*option* ...]

Description

The `uerf` command prints a record of system events. These events include error messages relating to the system hardware and the software kernel as well as information about system status, startup, and diagnostics.

Options

–c *classes* Selects classes of events.

err	Reports all hardware- and software-detected errors.
maint	Reports any event that occurs during system maintenance, for example, running the on-line functional exercisers.
oper	Reports information on system status, autoconfiguration messages, device status or error messages, time stamps, and system startup and shutdown messages.

–D *disks* Selects the disk device type. If you do not specify any parameters, all disk types are reported.

–f *filename* Outputs error information from the specified file rather than the default errorlog file `/usr/adm/syserr/syserr.`*hostname*. The *hostname* argument is the name of the local system. Use this option to look at old or backup errorlog files not in the default location `/usr/adm/syserr`. You can also use this option to access the default single-user errorlog file `/syserr.`*hostname*. Specify the full path name for the file. Do not use the `-n` option with this option.

–h Displays a brief help message. If you specify any other option with the `-h` option, it is ignored.

–H *host* Selects errors for the system indicated. Use the `-H` option when you are logging errors from multiple remote systems to a single errorlog file on the local host.

–M *mainframe_errors*
Selects mainframe error types. If you do not specify any parameters, all error types are reported.

cpu	Reports CPU-related errors such as machine checks.
mem	Reports memory-related errors such as single-bit CRD (corrected read data) and double-bit uncorrectable errors.

–n Displays errors as they occur in real time before logging them in the errorlog file. This option is useful in monitoring errors while a disk or tape exerciser is run. You cannot use this option with the `-f` option.

–o *output* Outputs errors in brief, full, or terse format. The default output is brief.

brief	Reports error information in a short format.
full	Reports all available information for each entry.
terse	Reports error information and displays register values, but does not translate.

–O *operating_system_events*
Selects operating system events such as panics and exceptions and faults. If you do not specify any parameters, all ULTRIX operating system events are reported. Reports are as follows:

aef	Arithmetic exception faults
ast	Asynchronous trap exception faults
bpt	Breakpoint instruction faults
cmp	Compatibility mode faults
pag	Page faults
pif	Privileged instruction faults
pro	Protection faults
ptf	Page table faults
raf	Reserved address faults
rof	Reserved operand faults
scf	System call exception faults
seg	Segmentation faults
tra	Trace exception faults
xfc	Reports xfc instruction faults

–R *reverse chronological order*
Outputs selected error information in reverse chronological order.

–r *records* Reports errors for the selected record codes. Valid codes are:

Hardware-Detected Errors

100	Machine check
101	Memory corrected read data/read data substitute (crd/rds)
102	Disk errors
103	Tape errors
104	Device controller errors
106	Bus errors
107	Stray interrupts
108	Asynchronous write errors
109	Exceptions/faults
112	Stack dump

Software Detected Errors

200	Panics (bug checks)

Informational ASCII Messages

250	Informational

Operational Messages

300	Start up
301	Shutdown
310	Time change
350	Diagnostic information
351	Repair information

–s *sequence_numbers*
Reports errors for selected sequence numbers. When used by itself, this option will give all records with specified sequence numbers in the file.

–t *time_range* Selects errors for the specified time range. Without the `-t` option, the `uerf` command processes the errorlog file from beginning to end. A start date or time or an end date or time must be specified with the **–t** option. For partial entries, the default date is the current date, the default start time is 00:00:00, and the default end time is 23:59:59. The format is as follows:

uerf –t s:dd-mmm-yyyy,hh:mm:ss e:dd-mmm-yyyy,hh:mm:ss

Where:

s	Specifies the start date and time
e	Specifies the end date and time
dd	Day
mmm	Month
yyyy	Year
hh	Hour
mm	Minute
ss	Second

–T *tapes* Selects the tape device type. If you do not specify any parameters, all tape types are reported.

–x Excludes specified selection options from the report, whether they appear before or after the option. This option does not affect the `-f, -h, -H, -n, -o, -R, -t` options.

–Z Displays the entry in hexadecimal format.

Restrictions

The `uerf` command uses the data files `uerf.bin`, `uerf.hlp`, and `uerf.err`. The `uerf.bin` file is the event information database, the `uerf.hlp` file is the help file, and the `uerf.err` file is the error message file.

The `uerf` command searches for the `uerf` data files as follows:

1. If `uerf` is invoked with a full pathname, the `uerf` first checks that directory for the `uerf` data files.
2. Then the `/etc` directory is checked.
3. And finally, directories specified in the Shell PATH environment variable are checked.

The `uerf` command outputs the contents of the errorlog file in the directory specified in `/etc/elcsd.conf`. To report on any other errorlog file, such as the single-user errorlog file, you must use `uerf` with the `-f` option.

Do not specify any other option with the `-h` option.

You cannot use the `-n` option and the `-f` option together.

Some hardware and system-related errors are logged as ASCII informational messages. Use the `-r` option with record type 250 to output these errors.

Examples

The following example produces a report containing all error events excluding logged operating system errors and operator and maintenance class errors:

```
uerf -O -x -c oper,maint
```

The following example produces an error report from the named file:

```
uerf -f /usr/adm/syslog/olderrorfile
```

The following examples show how to produce error reports for specific record codes:

```
uerf -r 100,102

uerf -r 100-109
```

The following examples show how to produce error reports using the `-t` option. This example lists all errors between 10:47 a.m. on April 13, 1990 and 5:30 p.m. on April 20, 1990.

```
uerf -t s:13-apr-1990,10:47:00 e:20-apr-1990,17:30:00
```

The following example produces an error report for all logged errors on the current day and year, which starts at 1:20 p.m. and ends at the current time.

```
uerf -t s:13:20
```

The next example produces an error report for all logged errors and displays it in reverse chronological order, starting with the current date and time.

```
uerf -R
```

Files

`/usr/adm/syserr/syserr.`*system-name*	Multiuser default errorlog file
`/etc/uerf.err`	The `uerf` error message file
`/etc/uerf.hlp`	The `uerf` help file
`/etc/uerf.bin`	Event information database file

See Also

elcsd.conf(5), elcsd(8), eli(8)
Guide to the Error Logger System

Name

uerf – ULTRIX error report formatter

Syntax

/etc/uerf [*option* ...]

Description

The `uerf` command prints a record of system events. These events include error messages relating to the system hardware and the software kernel as well as information about system status, startup, and diagnostics.

Options

-A *adapter* ... Selects adapter and device controller errors. All adapter error types are selected if none is specified.

aie	Reports errors for BVP-type controller.
aio	Reports errors for BVP-type controller.
bla	Reports errors for the BI LESI adapter.
bua	Reports errors for the BI UNIBUS adapter.
nmi	Reports errors for the Nautilus memory interconnect.
uba	Reports errors for the VAX UNIBUS adapter.

-c *classes* Selects classes of events.

err	Reports all hardware- and software-detected errors.
maint	Reports any event that occurs during system maintenance, for example, running the on-line functional exercisers.
oper	Reports information on system status, autoconfiguration messages, device status or error messages, time stamps, and system startup and shutdown messages.

-D *disks* Selects the mscp or scsi disk device type, for example, ra60 or rx23, or class, for example, ra or rx, about which to report errors. If you do not specify any parameters, all disk types are reported. See `ra`(4) for a list of supported

-f *filename* Outputs error information from the specified file rather than the default errorlog file `/usr/adm/syserr/syserr.`*hostname*. The *hostname* argument is the name of the local system. Use this option to look at old or backup errorlog files not in the default location `/usr/adm/syserr`. You can also use this option to access the default single-user errorlog file `/syserr.`*hostname*. Specify the full path name for the file. Do not use the `-n` option with this option.

-h Displays a brief help message. If you specify any other option with the `-h` option, it is ignored.

-H *host* Selects errors for the system indicated. Use the `-H` option when you are logging errors from multiple remote systems to a single errorlog file on the local host.

-M *mainframe_errors*
Selects mainframe error types. If you do not specify any parameters, all error types are reported.

cpu	Reports CPU-related errors such as machine checks.
mem	Reports memory-related errors such as single-bit CRD (corrected read data) and double-bit uncorrectable errors.

-n Displays errors as they occur in real time before logging them in the errorlog file. This option is useful in monitoring errors while a disk or tape exerciser is run. You cannot use this option with the **-f** option.

-o *output* Outputs errors in brief, full, or terse format. The default output is brief.

brief	Reports error information in a short format.
full	Reports all available information for each entry.
terse	Reports error information and displays register values, but does not translate.

-O *operating_system_events*
Selects operating system events such as panics and exceptions and faults. If you do not specify any parameters, all ULTRIX operating system events are reported. Reports are as follows:

aef	Arithmetic exception faults
ast	Asynchronous trap exception faults
bpt	Breakpoint instruction faults
cmp	Compatibility mode faults
pag	Page faults
pif	Privileged instruction faults
pro	Protection faults
ptf	Page table faults
raf	Reserved address faults
rof	Reserved operand faults
scf	System call exception faults
seg	Segmentation faults
tra	Trace exception faults
xfc	Reports xfc instruction faults

-R *reverse chronological order*
Outputs selected error information in reverse chronological order.

-r *records* Reports errors for the selected record codes. Valid codes are:

Hardware-Detected Errors

100	Machine check
101	Memory corrected read data/read data substitute (crd/rds)
102	Disk errors
103	Tape errors
104	Device controller errors
105	Adapter errors
106	Bus errors
107	Stray interrupts
108	Asynchronous write errors
109	Exceptions/faults
112	Stack dump
113	ka650 error and status
114	6200 vector 60
115	6200 vector 54
116	ka420 error and status registers
117	ka3100 error and status registers
118	6400 vector 60
119	6400 vector 54
120	ka60 mbus error
130	General Error and status registers
200	Panics (bug checks)
201	ci ppd info
202	scs events

Informational ASCII Messages

250	Informational
251	8600/8650 snapshot taken

Operational Messages

300	Start up
301	Shutdown
310	Time change
350	Diagnostic information
351	Repair information

-s *sequence_numbers*
Reports errors for selected sequence numbers. When used by itself, this option will give all records with specified sequence numbers in the file.

–S Produces a summary report of selected events.

-t *time_range* Selects errors for the specified time range. Without the `-t` option, the `uerf` command processes the errorlog file from beginning to end. A start date or time or an end date or time must be specified with the `-t` option. For partial entries, the default date is the current date, the default start time is 00:00:00, and the default end time is 23:59:59. The format is as follows:

uerf –t s:dd-mmm-yyyy,hh:mm:ss e:dd-mmm-yyyy,hh:mm:ss

Where:

s	Specifies the start date and time
e	Specifies the end date and time
dd	Day
mmm	Month
yyyy	Year
hh	Hour
mm	Minute
ss	Second

-T *tapes* Selects the tmscp or scsi tape types (tk50 or tz30, for example) or class (tk or tz, for example) for which to report errors. If you do not specify any parameters, all tape types are reported. See `tms`(4) for a list of supported tmscp tape types.

-x Excludes specified selection options from the report, whether they appear before or after the option. This option does not affect the **–f, –h, –H, –n, –o, –R, –t** options.

-Z Displays the entry in hex format.

Restrictions

The `uerf` command uses the data files `uerf.bin`, `uerf.hlp`, and `uerf.err`. The `uerf.bin` file is the event information database; the `uerf.hlp` file is the help file; and the `uerf.err` file is the error message file.

The `uerf` command searches for the `uerf` data files as follows:

1. If `uerf` is invoked with a full pathname, the `uerf` first checks that directory for the `uerf` data files.
2. Then the `/etc` directory is checked.
3. And finally, directories specified in the Shell PATH environment variable are checked.

The `uerf` command outputs the contents of the errorlog file in the directory specified in `/etc/elcsd.conf`. To report on any other errorlog file, such as the single-user errorlog file, you must use `uerf` with the `-f` option.

Do not specify any other option with the `-h` option.

You cannot use the `-n` option and the `-f` option together.

Some hardware and system-related errors are logged as ASCII informational messages, for example, MASSBUS device errors and UNIBUS communication device errors. Use the `-r` option with record type 250 to output these errors.

Examples

The following example produces a report containing all uba and nmi errors:

```
uerf -A uba,nmi
```

The following example produces a report containing all error events excluding logged operating system errors and operator and maintenance class errors:

```
uerf -O -x -c oper,maint
```

The following example produces an error report from the named file:

```
uerf -f /usr/adm/syslog/olderrorfile
```

The following examples show how to produce error reports for specific record codes:

```
uerf -r 100,102
```

```
uerf -r 100-109
```

The following examples show how to produce error reports using the `-t` option. This example lists all errors between 10:47 a.m. on April 13, 1990 and 5:30 p.m. on April 20, 1990.

```
uerf -t s:13-apr-1990,10:47:00 e:20-apr-1990,17:30:00
```

The following example produces an error report for all logged errors on the current day and year, which starts at 1:20 p.m. and ends at the current time.

```
uerf -t s:13:20
```

The next example produces an error report for all logged errors and displays it in reverse chronological order, starting with the current date and time.

```
uerf -R
```

Files

`/usr/adm/syserr/syserr.`*system-name*	Multiuser default errorlog file
`/etc/uerf.err`	The `uerf` error message file
`/etc/uerf.hlp`	The `uerf` help file
`/etc/uerf.bin`	Event information database file

See Also

elcsd.conf(5), elcsd(8), eli(8)
Guide to the Error Logger System

update (8)

Name

update – periodically update the super block

Syntax

/etc/update

Description

The `update` program executes the `sync`(2) primitive every 30 seconds. This insures that the file system is fairly up to date in case of a crash. This command should not be executed directly. Rather, it should be executed out of the initialization shell command file.

Restrictions

With `update` running, if the CPU is halted just as the `sync` is executed, a file system can be damaged. This is partially due to Digital hardware that writes zeros when NPR requests fail. A fix would be to have `sync(1)` temporarily increment the system time by at least 30 seconds to trigger the execution of `update`. This would give 30 seconds grace to halt the CPU.

See Also

sync(1), sync(2), init(8), rc(8),

Name

uucompact, uumkspool, uurespool, uupoll – uucp utilities

Syntax

uucompact –s*system*
uumkspool *system ...*
uurespool [**–t***#*]
uupoll *system ...*

Description

All of the `uucp` commands are located in `/usr/lib/uucp`.

The `uucompact` command compacts uucp system spool directories and associated subdirectories. If *system* is ALL, then all existing uucp system spool directories are compacted. Otherwise, only the specified system spool directory is compacted. If no system is specified, `/usr/spool/uucp/sys` is compacted. If `uucompact` is stopped before it is finished, it can be restarted without reprocessing directories. The `uucompact` command continues processing where it left off during it's previous instantiation.

The `uumkspool` command makes a per system spool directory and associated subdirectories for each of the specified systems. For example, if *system* is mk3 and if the local system name is penny, the following directories are created:

```
/usr/spool/uucp/sys/mk3
/usr/spool/uucp/sys/mk3/C.
/usr/spool/uucp/sys/mk3/X.
/usr/spool/uucp/sys/mk3/D.
/usr/spool/uucp/sys/mk3/D.penny
/usr/spool/uucp/sys/mk3/D.penny
```

The `uurespool` command moves files from old spool directories to new spool directories. Because the structure of the spool directories has changed from older versions of `uucp,` it is necessary to respool old spooled files to new spool directories in at least two instances:

- When installing the current version of `uucp.`
- When creating a new system spool directory for each system.

In the latter case, it is necessary to move files from `/usr/spool/uucp/sys/DEFAULT` to the new spool directories. To ease this task, `uurespool` moves files that have been spooled in one of 4 formats and respools them under the new spooling structure. The format is specified by the `-t#` option, where the number sign (#) can be any one of the following:

- Original spool - All files are in `/usr/spool/uucp.`
- Split spool - Contains the subdirectories `C., X., D., D.local, D.localX.`
- Modified split spool - Contains all subdirectories listed in split spool, and `STST., TM., C./OTHERS.`
- Used when a new system directory has been created and spool files must be moved from the DEFAULT directory to the new system directory.

The `uupoll` command forces a connect attempt to the named systems even if recent attempts have failed, but not if the `L.sys` file prohibits the call. For example, the `L.sys` file will prohibit the call if it is the wrong time of day. Thus, the `/usr/spool/uucp/LOGFILE` should be monitored for messages about the connection.

Files

`/usr/spool/uucp`	Spool directory
`/usr/spool/uucp/LOGFILE`	Logfile

See Also

mail(1), uucp(1c), uux(1c)

Name

uucico – uucp file transfer daemon

Syntax

/usr/lib/uucp/uucico [**–f**] [**–r1**] [**–s** *system*] [**–x#**] [**–X#**]

Description

The `uucico` program is the daemon associated with the `uucp` utility. The `uucico` daemon is automatically run when `uucp` requests are made or when polling occurs as specified in `/etc/crontab`. It performs the actual file transfer over the network. The `uucico` program first calls up the destination system and login. After a successful login, a handshake takes place between the destination `uucico` daemon process and the local daemon. The handshake determines if each daemon has permission to use the other's local resources. Both daemons then select the protocol that will be used to send and receive raw data.

The local daemon searches the spool directories for job requests, builds a list of files to transfer, then begins transmitting files. The file transfer protocol ensures that each file is transmitted only once and also notifies the user if a file cannot be transferred. After the `uucico` program transfers all the files, the destination site transfers files back to the local system. When both systems have completed their file transfers, the connection through the network is terminated.

The output of `uucico` follows the progress of the conversation. Debugging output from the slave `uucico` is placed in the file `AUDIT`, in the spool directory at the remote site. The ouput is less meaningful than the `LOGFILE`, unless the source code is available to help interpret the messages.

Options

-f Forces `uucico` to start a conversation with a specified system, regardless of any previous connection status as provided by the `STST.` files (system status).

-r1 Puts `uucico` into the master role. The slave mode is the default.

-s *system*
Designates the system to be contacted.

-x# Sets the debugging level. The number sign (#) can have a value of 0 to 9. The higher the number, the more debugging output. No packet level debugging is printed.

-X# Sets packet level debugging output. The number sign (#) can have a value of 0 to 9. The higher the number, the more packet level debugging output.

Files

```
/usr/spool/uucp
/usr/spool/uucp/STST.
/usr/spool/uucp/LOGFILE
/usr/spool/uucp/AUDIT
/usr/lib/uucp/uucp.*
```

See Also

uucp(1c), uux(1c)
Guide to the uucp Utility

Name

uuclean – uucp spool directory clean-up

Syntax

uuclean –p[*pre*] [*options* ...]

Description

The `uuclean` command scans the spool directory for files with the specified prefix and deletes all those which are older than the specified number of hours.

The `-p`*pre* argument causes the `uuclean` command to scan for files with *pre* as the file prefix. You can specify up to 10 `-p` arguments. A `-p` without any *pre* following causes all files older than the specified time to be deleted. You must specify at least one `-p` argument.

Options

–n*time*	Delete all files whose age is more than *time*, in hours, (default is 72 hours) and that have the specified *pre* as their file prefix.
–m	Send mail to the owner of the file when it is deleted.
–s*system*	Delete files in all directories that are subdirectories of the per system spool directory that exists for *system*. If ALL is specified, then all system directories are processed. ALL is the default.
–d*directory*	Delete files that reside in the named *directory*. The default directory is `/usr/spool/uucp`. The `-s` option over rides the `-d` option.

The `cron` command typically starts the `uuclean` program. In earlier versions, a deleted work file (C.file) would result in mail to the owner of the work file, regardless of the `-m` option. Now, notification of deleted work files is sent to the user ID "uucp". If the `-m` option is used, mail is also sent to the owner.

Examples

Here are some example `uuclean` command lines:

```
# uuclean -pLTMP. -pLOG. -n4 -d/usr/spool/uucp
# uuclean -d/usr/spool/uucp/.XQTDIR -p -n
# uuclean -smarkie -p -n84
```

The third example deletes all `uucp` files for the system `markie` that are older than 84 hours.

Files

`/usr/lib/uucp`	Directory with commands used by uuclean internally

See Also

uucp(1c), uux(1c), cron(8)

uucpsetup (8)

Name

uucpsetup – uucp set up program

Syntax

/etc/uucpsetup [–moia]

Description

The `uucpsetup` command provides an interactive facility for setting up and updating the `uucp` files necessary to configure your system for `uucp` connections.

To set up `uucp` initially, run the `uucpsetup` command with the `-a` option and answer the questions. To update the `uucp` files and directories, run `uucpsetup` with the `-m`, `-o`, and/or `-i` options, as appropriate. If no options are specified, the `-o` and `-i` options are the default.

You must be superuser to run `uucpsetup`.

Options

–m	Adds modems, configuring the dialers for `uucp` and `tip`.
–i	Adds incoming systems; that is, systems that are allowed to call your local system.
–o	Adds outgoing systems; that is, systems that your local system is allowed to call.
–a	Adds the modems and incoming and outgoing systems to `uucp`. The `-a` option implies the `-m`, `-i`, and `-o` options.

Files

`/usr/lib/uucp/L.sys`	Systems called out to
`/usr/lib/uucp/L-devices`	Device information
`/usr/lib/uucp/USERFILE`	Systems which call in
`/usr/lib/uucp/L.cmds`	Command execution protections

See Also

Guide to the uucp Utility

Name

uumonitor – monitor the UUCP system

Syntax

uumonitor

Description

The `uumonitor` command displays a synopsis in tabular format of the current UUCP status. The format of each line is as follows:

system_name #C #X *most_recent_status* CNT:# *time*

In the format of this line,

system_name	is the remote system for which the entry applies.
#C	is the number of C.files queued for the remote system.
#X	is the number of requests for remote execution from the remote system.
most_recent_status	is the result of the most recent attempt to connect to the remote system.
CNT:#	is the number of times that a failure to log in to the remote system has occurred. This does NOT include the number failed dial attempts.
time	is the time of the last status entry was made for this system.

The `uumonitor` command is helpful for detecting systems that have backlogs, that have gone away for awhile, that have changed phone numbers, and so forth. The CNT: field is useful for detecting a system whose login/passwd has changed. If the CNT: field gets larger than the maximum allowable failures (currently 20), no further attempts to connect to this system are made. If the number of C.files queued starts getting unusually large (depending on the system anywhere from 100-1000), action should be taken to determine the cause of the backlog.

See Also

uucp(1c)

Name

vipw – edit the password file

Syntax

vipw

Description

The `vipw` command edits the password file while setting the appropriate locks and doing any necessary processing after the password file is unlocked. If the password file is already being edited, then you will be told to try again later. The `vi` editor will be used unless the environment variable EDITOR indicates an alternate editor. The `vipw` command performs a number of consistency checks on the password entry for root, and does not allow a password file with a corrupted root entry to be installed. If the hashed `passwd` database exists `vipw` will automatically invoke `mkpasswd`(8) on it to keep it synchronized with the `passwd` file.

Files

`/etc/ptmp`

See Also

chfn(1), chsh(1), passwd(1), passwd(5yp), adduser(8), mkpasswd(8)

Name

xf – transparent filter

Syntax

/usr/lib/lpdfilters/xf

Description

The `xf` filter is a transparent filter. The filter simply concatenates data directly to the printer. The `xf` filter may be used with any printer. This filter allows device specific escape characters and control sequences to reach the printer.

There are no command line arguments passed to the `xf` filter by `lpd` and the filter does not maintain any accounting information.

The `xf` filter should be specified in the **xf** field of the `/etc/printcap` file. For further information, see `printcap`(5). It is invoked when the **–x** option is used with the `lpr`(1) command to print a job. Prior to its invocation the filter specified in the **of** field is invoked to print the banner page upon completion it is stopped and the `xf` filter sends the job to the printer.

Files

```
/etc/printcap
/dev/lp?
```

See Also

lpr(1), pr(1), printcap(5), lpd(8), MAKEDEV(8), pac(8)
''Line Printer Spooler Manual,'' *ULTRIX Supplementary Documents, Volume 2: Programmer*

xlator_call (8)

Name

xlator_call – shell script to invoke PostScript (TM) translators

Syntax

xlator_call *datatype orientation pagesize width length indent*

Description

This Bourne shell script can be called by the `lpd` line printer daemon to invoke the appropriate translator to convert each data type to PostScript (TM). The data type passed by `lpd` may be specified by using the `lpr` command with the **–D***datatype* option, or by using the **Da=datatype** capability in the `printcap` file.

For a file in a data syntax to be translated to PostScript (TM), the `xlator_call` script must contain a case branch which recognises the data type string and calls a suitable filter. The supplied `xlator_call` recognises the ANSI, ASCII, Regis and Tektronix 4014 data types.

Arguments

datatype
The valid data types are: `ansi`, `ascii`, `postscript`, `regis`, `tek4014`, or any other for which a translator has been installed.

orientation
The valid orientations are: *portrait* or *landscape*.

pagesize
The valid page sizes are: `letter`, `a`, `ledger`, `b`, `legal`, `executive`, `a5`, `a4`, `a3`, `b5`, or `b4`.

width
The width (in characters) of the page.

length
The length (in lines) of the page.

indent
The amount (in spaces) the output is to be indented.

All the valid arguments are described in detail by the `lpr(1)` reference page.

Examples

An example shell script is shown below:

```
case $datatype in
ansi)
       exec ansi_ps -F $pagesize -O $orientation -e "$@";;
ascii)
       echo "(\004) cvn {} def"
       exec ln03rof -w$width -l$length -i$indent;;
postscript)
       exec cat;;
tek4014)
       exec tek4014_ps -F $pagesize -O $orientation;;
regis)
       exec regis_ps -F $pagesize -O $orientation;;
```

```
*)
      echo "$0: Translator for data type $datatype not installed" >&2
esac
```

When the shell script is called, the path searched is:

```
/usr/local/lib/lpdfilters:/usr/ucb:/bin:/usr/bin:
                                /usr/lib:/usr/lib/lpdfilters
```

Files

`/usr/lib/lpdfilters/xlator_call`	The `xlator_call` script

See Also

lno3rof(8), printcap(5), ansi_ps(8), lpd(8)

ypmake (8yp)

Name

ypmake – rebuild yellow pages (YP) database using the make command

Syntax

cd /etc/yp
make [*options*] [*map*]

Description

The `make` command uses the `/etc/yp/Makefile` to build the yellow pages database. With no arguments, `make` creates `dbm` databases for any YP maps that are out-of-date, and then executes `yppush`(8yp) to notify slave databases that there has been a change. It is important to note that the `ypmake` command should only be executed at a YP master server machine. If it is executed from either a slave server or a pure YP client machine, the created changes will only be overwritten when the next YP master server machine update, using `ypxfr`, is performed.

The *options* argument can be used to change the default values of three special variables used by `make`: *DIR* , *NOPUSH* , and *DOM* . The *DIR* variable instructs `make` to give the directory of the source files. The *NOPUSH* variable, which when non-null, inhibits updating of the new data base files using the `yppush`(8yp) function. The *DOM* variable, instructs `make` to construct a domain other than the master's default domain. The default for *DIR* is `/etc`, and the default for *NOPUSH* is the null string. To change the default values of these special variables, an *options* argument format of *special_variable = value* is used. See the Examples section for an example.

The *map* argument supplied on the command line instructs `make` to update only the specified map. The specified maps are those located at `/etc/yp/{domain}`, where {domain} is the yellow pages domain name. Some typical entries for the *map* argument are passwd, hosts, and networks. Typing `make passwd` causes `make` to create and update the YP password database, if it is out of date. The `make` command updates the password data base using `yppush`(8yp). Therefore, typing

```
make host
```

or

```
make networks
```

causes `make` to create and to update the host and network files, `/etc/hosts` and `/etc/networks` respectively.

See `ypfiles`(5yp) and `ypserv`(8yp) for an overview of the yellow pages.

Examples

This example causes `make` to create a password yellow pages map for the domain `NewDomain` instead of for the default domain:

```
make DOM=NewDomain passwd
```

See Also

make(1), makedbm(8yp), ypserv(8yp), yppush(8yp)

yppasswdd (8yp)

Name

yppasswdd – server daemon for modifying the yellow pages (YP) password file

Syntax

/usr/etc/rpc.yppasswdd *file* [**–m** *arg1 arg2* ...]

Description

The `yppasswdd` daemon is a server that handles password change requests from `yppasswd`(1yp). It changes a password entry in the specified *file,* which is assumed to be in the same format described in `passwd`(5yp). An entry in *file* will be changed only if the password presented by `yppasswd`(1yp) matches the encrypted password of that entry.

If the `-m` option is given, then after *file* is modified, a `make`(1) will be performed in `/var/yp`. Any arguments following the flag will be passed to `make`. The `-m` option should be set only at a YP master server machine.

This server is not run by default, nor can it be started up from `inetd`(8c). If it is desired to enable remote password updating for the yellow pages, then an entry for `yppasswdd` should be put in the `/etc/rc.local` file of the host serving as the master for the yellow pages `passwd` file.

Examples

If the yellow pages password file is stored as `/var/yp/src/passwd,` then to have password changes propagated immediately, the server should be invoked as:

```
/usr/etc/rpc.yppasswdd /var/yp/src/passwd -m passwd DIR= /var/yp/src
```

Files

`/var/yp/Makefile`

See Also

yppasswd(1yp), passwd(5yp), ypfiles(5yp), ypmake(8yp)

Name

yppoll – determine which version of a yellow pages (YP) map is at a master YP server host.

Syntax

yppoll [**-h** *host*] [**-d** *domain*] *mapname*

Description

The `yppoll` command asks a `ypserv` process what the order number is, and which host is the master YP server for the named map. If the server is a v.1 YP protocol server, `yppoll` uses the older protocol to communicate with `ypserv` and uses the older diagnostic messages if a failure occurs.

Options

–h *host* Ask the `ypserv` process at *host* about the map parameters. If *host* is not specified, the YP server for the local host is used. That is, the default host is the one returned by `ypwhich`(1yp).

–d *domain*
Use *domain* instead of the default domain.

See Also

ypfiles(5yp), ypserv(8yp)

yppush (8yp)

Name

yppush – force propagation of a changed yellow pages (YP) map

Syntax

yppush [**–d** *domain*] [**–v**] *mapname*

Description

The `yppush` command copies a new version of a yellow pages (YP) map from the master YP server to the slave YP servers. It is normally run only on the master YP server by the `make` utility accessing the `/var/yp/Makefile` after the master YP databases have been changed. When invoked, `yppush` first constructs a list of YP server hosts by reading the YP map *ypservers* within the *domain.* Keys within the map *ypservers* are the ASCII names of the machines on which the YP servers run.

A transfer map request is sent to the YP server at each host, along with the information needed by the transfer agent (the program which actually moves the map) to call back the `yppush` command. When the attempt has completed (successfully or not), and the transfer agent has sent `yppush` a status message, the results can be printed to stdout. Messages are also printed when a transfer is not possible, for instance when the request message is undeliverable, or when the timeout period on responses has expired.

Refer to `ypfiles`(5yp) and `ypserv`(8yp) for an overview of the yellow pages.

Options

–d	Specify a *domain.*
–v	Verbose. This causes messages to be printed when each server is called, and for each response. Without this flag, only error messages are printed.

Restrictions

In the current implementation (version 2 YP protocol), the transfer agent is `ypxfr`, which is started by the `ypserv` program. If `yppush` detects that it is speaking to a version 1 YP protocol server, it uses the older protocol, sending a version 1 YPPROC_GET request and issues a message to that effect. Unfortunately, there is no way of knowing if or when the map transfer was performed for version 1 servers. The `yppush` command prints a message saying that an old-style message has been sent. The system administrator should later check to see that the transfer has actually taken place.

Files

`/etc/yp/`*domainname*`/ypservers.{dir, pag}`

See Also

ypfiles(5yp), ypserv(8yp), ypxfr(8yp)

Name

ypserv, ypbind – yellow pages (YP) server and binder processes

Syntax

/usr/etc/ypserv
/etc/ypbind [–S] *domainname, servername*

Description

The yellow pages (YP) service provides a network lookup service consisting of databases and processes. The databases are `dbm(3x)` files stored in the `/etc/yp` directory. These files are described in `ypfiles`(5yp). The processes are `/usr/etc/ypserv`, the YP database lookup server, and `/etc/ypbind`, the YP binder. The software interface to the YP service is described in `ypclnt`(3yp). Administrative tools are described in `yppush`(8yp), `ypxfr`(8yp), `yppoll`(8yp), and `ypwhich`(1yp). Tools to see the contents of YP maps are described in `ypcat`(1yp), and `ypmatch`(1yp). Database generation and maintenance tools are described in `ypmake`(8yp), and `makedbm`(8yp).

Both `ypserv` and `ypbind` are daemon processes activated at system startup time from `/etc/rc.local`. The `ypserv` command runs only on a YP server machine with a complete YP database. The `ypbind` command runs on all machines using YP services, both YP servers and clients.

The `ypserv` daemon's primary function is to look up information in its local database of YP maps. The operations performed by `ypserv` are defined for the programmer in the header file `<rpcsvc/yp_prot.h>`.

Communication with `ypserv` is by means of RPC calls. Lookup functions are described in `ypclnt`(3yp), and are supplied as C-callable functions in `/lib/libc`.

There are four lookup functions, all of which are performed on a specified map within a YP domain: `Match`, `Get_first`, `Get_next`, and `Get_all`. The `Match` operation takes a key, and returns the associated value. The `Get_first` operation returns the first key-value pair from the map, and the `Get_next` operation returns the remaining key-value pairs. The `Get_all` operation ships the entire map to the requester.

Two other functions supply information about the map, rather than the map entries: `Get_order_number`, and `Get_master_name`. Both the order number and the master name exist in the map as key-value pairs, but the server will not return either through the usual lookup functions. If the map is examined with `makedbm`(8yp), however, they will be visible.

Other functions are used within the YP subsystem itself, and are not of general interest to YP clients. They include the `Do_you_serve_this_domain?`, the `Transfer_map`, and the `Reinitialize_internal_state` functions.

The purpose of the `ypbind` function is to remember information that lets client processes on a single node communicate with a `ypserv` process. The `ypbind` function *must* run on every machine that has YP client service requirements. The `ypbind` function must be started through an entry in the `/etc/rc.local` file. The `-S` option allows the system administrator to lock `ypbind` to a particular domain and set of servers. Up to four servers can be specified as follows:

```
/etc/ypbind -S domainname,server1,server2,server3,server4
```

The ypserv function may or may not be running on the same node, but must be running somewhere on the network. The servers used with the -S option must have entries in the local /etc/hosts file.

The information ypbind remembers is called a *binding*, the association of a domain name with the internet address of the YP server, and the port on that host at which the ypserv process is listening for service requests. The process of binding is driven by client requests. As a request for an unbound domain comes in, the ypbind process broadcasts on the net trying to find a ypserv process that serves maps within that domain. Since the binding is established by broadcasting, there must be at least one ypserv process on every net. Once a domain is bound by a particular ypbind, that same binding is given to every client process on the node. The ypbind process on the local node or a remote node may be queried for the binding of a particular domain by using the ypwhich(1yp) command.

Bindings are verified before they are given out to a client process. If ypbind is unable to speak to the ypserv process it is bound to, it marks the domain as unbound, tells the client process that the domain is unbound, and tries to bind the domain once again. Requests received for an unbound domain will fail immediately. In general, a bound domain is marked as unbound when the node running ypserv crashes or gets overloaded. When the node gets overloaded, ypbind will try to bind any YP server (typically one that is less-heavily loaded) available on the net.

The ypbind process also accepts requests to set its binding for a particular domain. The request is usually generated by the YP subsystem itself.

Files

If the file /var/yp/ypserv.log exists when ypserv starts up, log information will be written to ypserv.log when error conditions occur.

See Also

ypcat(1yp), ypmatch(1yp), ypwhich(1yp), ypclnt(3yp), ypfiles(5yp), yppush(8yp), ypxfr(8yp)

Name

ypsetup – set up the yellow pages (YP) environment

Syntax

/usr/etc/ypsetup

Description

A local area network must be set up on your system before you can set up YP. You must know your system's default YP domain name and whether your system will be a master server, slave server, or client. If your system is to be the master server for your YP domain, be sure no other master has been established. If your system is not to be the master server, be sure a master already exists for your YP domain. Once you know this information, run `ypsetup` with the system in multiuser mode and answer its questions.

Once YP is installed on a machine, it cannot be used until the `/etc/svc.conf` file is modified to contain YP entries on the desired database lines. The `ypsetup` command reminds a user to run `/usr/etc/svcsetup` or edit the `/etc/svc.conf` file manually.

Files

Files that Start the YP daemons

`/etc/crontab`	Clock daemon database file
`/etc/rc.local`	Commands pertinent to a specific system

Default YP Map Files

`group`	Group database
`hosts`	Host name database
`mail.aliases`	Sendmail alias database
`netgroup`	Network group aliases
`networks`	Network name database
`passwd`	Password file
`protocols`	Protocol name database
`rpc`	Rpc name database
`services`	Service name database

See Also

domainname(1yp), ypwhich(1yp), svc.conf(5), svcsetup(8), yppasswdd(8yp), ypserv(8yp), ypxfr(8yp)
Guide to the Yellow Pages Service

ypxfr (8yp)

Name

ypxfr – transfer a yellow pages (YP) map from a YP server to the local host.

Syntax

ypxfr [**–f**] [**-h** *host*] [**-d** *domain*] [**–c**] [**-C** *tid prog ipadd port*] *mapname*

Description

The `ypxfr` command moves a YP map, specified by the *mapname* argument, to the local host by making use of normal YP services. It creates a temporary map in the directory `/etc/yp/domain` (which must already exist), fills it by enumerating the map's entries, obtains the map parameters (master and order number) and loads them into the map. Once `ypxfr` has accomplished these tasks, it deletes any old versions of the map and moves the temporary map to the real mapname.

If `ypxfr` is run interactively, it writes its output to the terminal. However, if it is invoked without a controlling terminal, and if the log file `/etc/yp/ypxfr.log` exists, it will append all its output to that file. Since `ypxfr` is most often run from `/usr/lib/crontab,` or by `ypserv,` the log file can be used to retain a record of what was attempted, and the results.

For consistency between servers, `ypxfr` should be run periodically for every map in the YP database. Different maps change at different rates: the `services.byname` map may not change for months at a time, for instance, and may therefore be checked only once a day. It is possible that `mail.aliases` or `hosts.byname` changes several times per day. In such a case, it is appropriate to check hourly for updates. A `cron`(8) entry should be used to perform periodic updates automatically on YP server machines only. Rather than having a separate `cron` entry for each map, commands can be grouped to update several maps in a shell script. Examples (mnemonically named) are in `/etc/yp: ypxfr_1perday.sh, ypxfr_2perday.sh,` and `ypxfr_1perhour.sh.` They can serve as reasonable first cuts.

See `ypfiles`(5yp) and `ypserv`(8yp) for an overview of the yellow pages.

Options

–f Force the transfer to occur even if the version at the MASTER is not more recent than the local version.

–c Do not send a "Clear current map" request to the local `ypserv` process. This flag should be used if `ypserv` is not running locally at the time when `ypxfr` is running. Otherwise, `ypxfr` will report that it can not talk to the local `ypserv,` and the transfer will fail.

–h *host* Get the map from *host,* regardless of which map is the master. If *host* is not specified, `ypxfr` will ask the YP service for the name of the master, and try to get the map from there. The *host* option can be a name or an internet address in the form *a.b.c.d* .

–d *domain* Specify a domain other than the default domain.

–C *tid prog ipadd port*
: This option is only for use by `ypserv`. When `ypserv` invokes `ypxfr`, it specifies that `ypxfr` should call back a `yppush` process at the host with IP address *ipaddr*, registered as program number *prog*, listening on port *port*, and waiting for a response to transaction *tid*.

Files

```
/etc/yp/ypxfr.log
/etc/yp/ypxfr_1perday.sh
/etc/yp/ypxfr_2perday.sh
/etc/yp/ypxfr_1perhour.sh
/usr/lib/crontab
```

See Also

ypfiles(5yp), cron(8), yppush(8yp), ypserv(8yp), ypsetup(8yp)

zdump(8)

Name

zdump – time zone dumper

Syntax

zdump [**–c** *cutoffyear*] [**–v**] *zonename* ...

Description

The `zdump` command prints the current time in each *zonename* named on the command line.

Options

–c *cutoffyear*
Cut off the verbose output near the start of the *cutoffyear*.

–v For each *zonename* on the command line, prints the current time, the time at the lowest possible time value, the time one day after the lowest possible time value, the times both one second before and exactly at each time at which the rules for computing local time change, the time at the highest possible time value, and the time at one day less than the highest possible time value. Each line ends with isdst=1 if the given time is Daylight Saving Time or isdst=0 otherwise.

Files

`/etc/zoneinfo`
Standard zone information directory

See Also

ctime(3), tzfile(5), zic(8)

Name

zic – time zone compiler

Syntax

zic [**–d** *directory*] [**–l** *localtime*] [**–v**] [*filename* ...]

Description

The `zic` compiler reads text from the file(s) named on the command line and creates the time conversion information files specified in this input. If a *filename* is –, the standard input is read.

Input lines are made up of fields. Fields are separated from one another by any number of white space characters. Leading and trailing white space on input lines is ignored. An unquoted number sign (#) in the input introduces a comment which extends to the end of the line the sharp character appears on. White space characters and sharp characters may be enclosed in double quotation marks (" ") if they are to be used as part of a field. Any line that is blank (after comment stripping) is ignored. Non-blank lines are expected to be of one of three types: rule lines, zone lines, and link lines.

A rule line has the form

```
Rule  NAME  FROM  TO TYPE IN ON      AT    SAVE LETTER/S
```

For example:

```
Rule  USA   1969  1973 - Apr lastSun 2:00  1:00 D
```

The fields that make up a rule line are:

NAME	Gives the (arbitrary) name of the set of rules this rule is part of.
FROM	Gives the first year in which the rule applies. The word minimum (or an abbreviation) means the minimum year with a representable time value. The word maximum (or an abbreviation) means the maximum year with a representable time value.
TO	Gives the final year in which the rule applies. In addition to minimum and maximum (as above), the word only (or an abbreviation) may be used to repeat the value of the **FROM** field.
TYPE	Gives the type of year in which the rule applies. If **TYPE** is – then the rule applies in all years between **FROM** and **TO** inclusive; if **TYPE** is 'uspres', the rule applies in U.S. Presidential election years; if **TYPE** is 'nonpres', the rule applies in years other than U.S. Presidential election years. If **TYPE** is something else, then `zic` executes the following command: **yearistype** *year type* to check the type of a year: an exit status of zero is taken to mean that the year is of the given type; an exit status of one is taken to mean that the year is not of the given type.
IN	Names the month in which the rule takes effect. Month names may be abbreviated.

ON Gives the day on which the rule takes effect. Recognized forms include:

5	the fifth of the month
lastSun	the last Sunday in the month
lastMon	the last Monday in the month
Sun>=8	first Sunday on or after the eighth
Sun<=25	last Sunday on or before the 25th

Names of days of the week may be abbreviated or spelled out in full. Note that there must be no spaces within the **ON** field.

AT Gives the time of day at which the rule takes effect. Recognized forms include:

2	time in hours
2:00	time in hours and minutes
15:00	24-hour format time (for times after noon)
1:28:14	time in hours, minutes, and seconds

Any of these forms may be followed by the letter **w** if the given time is local 'wall clock' time or **s** if the given time is local 'standard' time; in the absence of **w** or **s**, 'wall clock' time is assumed.

SAVE Gives the amount of time to be added to local standard time when the rule is in effect. This field has the same format as the **AT** field (although, of course, the **w** and **s** suffixes are not used).

LETTER/S
Gives the 'variable part' (for example, the 'S' or 'D' in 'EST' or 'EDT') of time zone abbreviations to be used when this rule is in effect. If this field is –, the variable part is null.

A zone line has the form

```
"Zone NAME                  GMTOFF  RULES/SAVE FORMAT UNTIL]"
```

For example:

```
Zone  Australia/South-west 9:30    Aus         CST    1987 Mar 15 2:00
```

The fields that make up a zone line are:

NAME The name of the time zone. This is the name used in creating the time conversion information file for the zone.

GMTOFF
The amount of time to add to GMT to get standard time in this zone. This field has the same format as the **AT** and **SAVE** fields of rule lines; begin the field with a minus sign if time must be subtracted from GMT.

RULES/SAVE
The name of the rule(s) that apply in the time zone or, alternately, an amount of time to add to local standard time. If this field is – then standard time always applies in the time zone.

FORMAT
The format for time zone abbreviations in this time zone. The pair of characters **%s** is used to show where the variable part of the time zone abbreviation goes.

UNTIL The time at which the GMT offset or the rule(s) change for a location. It is specified as a year, a month, a day, and a time of day. If this is specified, the time zone information is generated from the given GMT offset and rule change until the time specified.

The next line must be a 'continuation' line; this has the same form as a zone line except that the string 'Zone' and the name are omitted, as the continuation line will place information starting at the time specified as the **UNTIL** field in the previous line in the file used by the previous line. Continuation lines may contain an **UNTIL** field, just as zone lines do, indicating that the next line is a further continuation.

A link line has the form

```
"Link    LINK-FROM    LINK-TO"
```

For example:

```
 Link    US/Eastern    EST5EDT
```

The **LINK-FROM** field should appear as the **NAME** field in some zone line; the **LINK-TO** field is used as an alternate name for that zone.

Except for continuation lines, lines may appear in any order in the input.

NOTE

For areas with more than two types of local time, you may need to use local standard time in the **AT** field of the earliest transition time's rule to ensure that the earliest transition time recorded in the compiled file is correct.

Options

–d *directory*
Create time conversion information files in the named *directory* rather than in the standard directory named below.

–l *timezone*
Use the given time zone as local time. The `zic` compiler will act as if the file contained a link line of the form:

Link   *timezone*`localtime`

–v Complain if a year that appears in a data file is outside the range of years representable by `time` values.

Files

`/etc/zoneinfo`   Standard directory used for created files

See Also

ctime(3), tzfile(5), zdump(8)

Index

Numbers

2780d daemon, 8–2

A

abort command (lpc), 8–188

ac program, 8–4

accounting file

maintaining, 8–326

accton command, 8–326

active gateway, 8–310

addgroup command, 8–7

addnode command, 8–5

See also getnode command

See also remnode command

options, 8–5

adduser command, 8–7

See also edauth editor

See also vipw command

aliases

conflicts, 8–58

ap program, 8–10

archive file

converting to portable format, 8–12

restoring on console diskette, 8–13 to 8–17

saving on console diskette, 8–13 to 8–17

arcv command, 8–12

arff command, 8–13 to 8–17, 8–16e

key modifiers, 8–14

keys, 8–13

specifying RT-11 file names, 8–15

arp keyword

See arp program

arp program, 8–18

audgen command, 8–19

audit daemon, 8–20

auditd command

options, 8–20

auditlog reduction tool, 8–24

auditmask command, 8–23

audit_tool command, 8–24

examples, 8–27

options, 8–24

auth database (commands)

maintenance, 8–116

auth entry

editing, 8–86

automount command, 8–28

options, 8–31

automount maps, 8–28

and Yellow Pages, 8–29

B

bad sector

creating file for, 8–35

reporting, 8–33

bad ssector

structure, 8–33e

bad144 command, 8–33

See also badsect program

badsect program, 8–35

BIND environment

setting up, 8–37

bindsetup command, 8–37

options, 8–37

biod daemon, 8–39

booting diskless clients, 8–40

bootstrap procedure, 8–291

automatic, 8–286

C

catman command, 8–42

ccr command, 8–43 to 8–44

diagnostics, 8–44

options, 8–43

restricted, 8–44

chown command, 8–45

chpt command, 8–47 to 8–49

options, 8–47

restricted, 8–49

clean command (lpc), 8–188

clock daemon, 8–71

clri command, 8–50

See also fsck command

cmx exerciser, 8–51, 8–52e

devices supported, 8–52

options, 8–51

pseudodevices and, 8–51

communications server

remote user, 8–252

comsat program, 8–53

config program

See also doconfig program, 8–54, 8–56

configuration file (syslog)

building, 8–54, 8–56, 8–76

format, 8–363

conflicts

aliases, 8–58

passwords, 8–58

console carrier requester

establishing remote connection, 8–43 to 8–44

console disk

building, 8–211

updating, 8–211

console diskette

copying, 8–96

core dump

saving, 8–328

crash dump

analyzing, 8–68, 8–70

crash keyword, 8–67, 8–69

crash utility, 8–59

cron command, 8–71

crontab file, 8–71

D

dcheck program

See also fsck command, 8–72

device

specifying for paging, 8–360

specifying for swapping, 8–360

dgated daemon, 8–73

Digital Storage Architecture

See DSA disk

disable command (lpc), 8–188

disk

testing, 8–79, 8–80

disk pack

formatting, 8–97 to 8–99

disk partition table

changing, 8–47 to 8–49, 8–48e

changing indiscriminately, 8–49c

changing procedure, 8–47

disk quota

checking, 8–280

editing, 8–88

enabling for file system, 8–281

summarizing by file system, 8–294

diskless client

booting, 8–40

dms program (DMS), 8–74

doconfig program, 8–76

See also sizer program

down-line load listener

prerequisites, 8–218

dp program, 8–78

drtest program, 8–79

running, 8–79e

DSA disk

managing, 8–282

dskx exerciser, 8–80
arguments, 8–80
examples, 8–81
options, 8–81
dump command, 8–82
See also rdump command
See also restore command
keys, 8–82
restrictions, 8–83
dumpfs command, 8–85

E

edauth editor, 8–86
edquota editor, 8–88
elcsd daemon, 8–89
See also uerf command
eli command, 8–90
enable command (lpc), 8–188
encryption key
generating, 8–207
error logging
initializing, 8–90
error logging daemon
See also error report formatter, 8–89
error report formatter, 8–383, 8–387
exit command (lpc), 8–188

F

file
See also archive file
changing owner, 8–45
removing directoryless, 8–50
sharing executable, 8–355, 8–356
file system, 8–222
See also NFS file system
See also UFS file system
changing layout policy, 8–381
checking consistency, 8–100 to 8–102, 8–127
checking directory consistency, 8–72
constructing prototype, 8–212, 8–216
copying across network, 8–288, 8–305
creating, 8–240, 8–242
file system (cont.)
incremental dumps, 8–82
listing file names, 8–236
mounting, 8–220
printing blocks per user, 8–279
printing disk usage, 8–294
printing information, 8–85
repairing interactively, 8–100 to 8–102
restoring incremental dumps, 8–295 to 8–298
restoring incremental dumps across network, 8–313 to 8–316
testing, 8–104
file transfers
uucico, 8–395
fingerd program, 8–93
flcopy command, 8–96
format program
disabling debugging, 8–98
ignoring diagnostics, 8–97, 8–97 to 8–99
running, 8–98e
specifying a drive, 8–97
fsck command
options, 8–102, 8–100 to 8–102
restricted, 8–102
fsirand command, 8–103
fsx exerciser
examples, 8–104
options, 8–104, 8–104
restricted, 8–104
ftpd server
options, 8–107, 8–106 to 8–108
request list, 8–106 to 8–107
restricted, 8–108
restricting access, 8–107
verifying users, 8–107
fverify command, 8–109

G

gateways file
creating, 8–126
entry format, 8–310
getauth command, 8–116

getnode command, 8–118
gettable program
  See also htable program, 8–119
getty program, 8–120
  diagnostics, 8–121
gprof command
  kgmon command, 8–150
group file
  adding groups, 8–7

H

halt command, 8–123, 8–124
help command (lpc), 8–188
hesupd
  Hesiod update daemon, 8–125
hosts file
  creating, 8–126
htable program
  See also getetable program, 8–126

I

icheck command, 8–127
  See also fsck command
  information reported, 8–127
  restricted, 8–128
ICMP ECHO_REQUEST, 8–268
ifconfig command
  arguments, 8–129
  description, 8–129
  diagnostics, 8–130
  interface parameter, 8–129
IMP log interpreter
  See implog program
implog file
  entry format, 8–132
implog program, 8–131
implogd program, 8–132
  See also implog program
inetd server daemon, 8–133
  See also miscd daemon
init program, 8–134, 8–136
  See also reboot command
init program (cont.)
  diagnostics, 8–135, 8–137
inode
  clearing, 8–50
Installing software subsets, 8–337
install-mh program, 8–138
Internet File Transfer Protocol server, 8–106
Internet service daemon, 8–133
intro(8) keyword, 8–1
IP network
  testing, 8–238 to 8–239, 8–238e

K

Kerberos routines
  ext_srvtab, 8–92
  kdb_destroy, 8–139
  kdb_edit, 8–140
  kdb_init, 8–141
  kdb_util, 8–143
  kdestroy, 8–145
  kerberos 8, 8–146
  kinit, 8–151
  klist, 8–153
  kprop, 8–155
  kpropd, 8–157
  kstash, 8–159
kgmon command, 8–150
kits
  subset fits on a system, 8–94

L

LAT protocol, 8–164
lcp command, 8–164
  error counter list, 8–167
  parameters, 8–164
License Management Facility commands, 8–173, 8–178
line printer
  setting up, 8–199
  testing, 8–200
line printer control program
  See lpc program

line printer daemon
See lpd daemon
lmf commands, 8–173, 8–178
load command
See also addnode command, 8–186
local area network
setting up files, 8–237
Local Area Transport protocol
See LAT protocol
Local Location Broker Daemon, 8–251, 8–172
lock
daemon, 8–187
login
accounting, 8–4
lost+found directory
creating, 8–213
lpc program, 8–188
commands, 8–188
lpd daemon
See also lpc program
key characters, 8–190, 8–190
lpq program
diagnostics, 8–189
lprsetup command, 8–199
lpx exerciser
examples, 8–201
options, 8–200, 8–200

M

magnetic tape
testing data, 8–229
mail
reporting incoming, 8–53
Maintenance Operation Protocol, 8–218
makedbm command
See also yppush command, 8–202
MAKEDEV command
making special files, 8–204, 8–205
makekey command, 8–207
manual page
formatting on line, 8–42
memory
testing, 8–208
memx
options, 8–208
memx exerciser
examples, 8–209, 8–208
restricted, 8–208
MH environment
initialization, 8–138
miscd daemon, 8–210
mkconsole program, 8–211
mkfs command, 8–212
See also mklost+found command
See also newfs command
See also tunefs command
mklost+found command, 8–213
mknod command, 8–214
mkpasswd command, 8–215
mkproto command
prototype file and, 8–216, 8–216
restricted, 8–216
mop_dumpload loader
prerequisites for load request, 8–218
mop_mom command, 8–218
See also load command
mount command, 8–226
mount command (general), 8–220
See also mount command (NFS)
See also ufs_mount command
restricted, 8–221
mount command (NFS), 8–222, 8–224
See also portmap daemon
See also ypserv daemon
options, 8–222
restricted, 8–224
mountd daemon, 8–222, 8–228
See also showmount command
mtx exerciser, 8–229
example, 8–230
options, 8–229

N

name
daemon, 8–231
named-xfer
named transfer daemon, 8–234
named-xfer transfer daemon
options, 8–234
ncheck command, 8–236
See also fsck command
netsetup command, 8–237
network
lock daemon, 8–187
Network File System
See NFS
network file transfers, 8–395
network interface
configuring parameters, 8–129
network status daemon, 8–354
networks file
creating, 8–126
netx exerciser, 8–238 to 8–239
options, 8–238
restricted, 8–238
newfs command, 8–240, 8–242
See also mkproto command
mkfs and, 8–240, 8–242
NFS
displaying statistics, 8–248
setting up, 8–247
NFS asynchronous block I/0 daemon
starting, 8–39
NFS file system, 8–222
showing remotely mounted, 8–344
NFS locking service
turning off the, 8–246
turning on the, 8–246
NFS mount request server, 8–228
nfsd daemon, 8–244
nfsportmon command, 8–245
nfssetlock command, 8–246
nfssetup command, 8–247
nfsstat command, 8–248
NIC standard host tables
converting, 8–126
getting, 8–119
node
changing data base entry, 8–5, 8–6e
displaying data base entry, 8–118, 8–118e
down-line loading, 8–186
removing from data base, 8–292
triggering down-line load request, 8–378, 8–378e
nologin file
contents, 8–345
Non-Replicatable Global Location Broker Daemon, 8–251
Non-replicating Global Location Broker Daemon
nrglbd, 8–251
ntalkd command, 8–252
ntpd daemon, 8–253
ntpdc command, 8–257

O

opser utility, 8–261

P

pac command, 8–262
packet filter
configure parameters, 8–263
pfconfig command, 8–263
pfstat command, 8–265
passive gateway, 8–310
password file
editing, 8–400
password file (system)
adding users, 8–7
removing users, 8–7
password file (YP)
modifying, 8–406
passwords
conflicts, 8–58
pfconfig command, 8–263
diagnostics, 8–264
pfstat command, 8–265

ping command, 8–268

plotter

getting paper consumption statistics, 8–262

port monitor

turning off the, 8–245

turning on the, 8–245

portmap daemon, 8–270

See also rpcinfo command

post program, 8–271

Postscript (TM) translator

datatype, 8–8

printer

getting paper consumption statistics, 8–262

process

changing priority of running, 8–293

initializing control, 8–134, 8–136

processor

halting, 8–123, 8–124

profile buffer

dumping, 8–150

pstat command, 8–273

restricted, 8–278

pstat program

options, 8–273

Q

quit command (lpc), 8–188

quot command, 8–279

quotacheck command, 8–280

See also edquota editor

See also quotaon command

See also repquota command

quotaoff command, 8–281

quotaon command, 8–281

quotas file

quotacheck command and, 8–280

R

radisk utility, 8–282

options, 8–282

random inode generation number

installing, 8–103

rarpd program

description, 8–284

options, 8–284

rc command, 8–286

rc.local file

contents, 8–286

rdate command, 8–287

rdump command

See also rmt program, 8–288

reboot command, 8–291

See also rc command

remnode command, 8–292

See also addnode command

remote login server

See rlogind server

remote mass storage protocol module

See rmt program

Remote Procedure Call

See RPC

remote shell server

See rshd server

removeuser command, 8–7

renice command, 8–293

repquota command, 8–294

restart command (lpc), 8–188

restore command, 8–295 to 8–298

See also rrestore command

diagnostics, 8–297 to 8–298

examples, 8–297

keys, 8–295 to 8–297

restricted, 8–297, 8–315

Reverse Address Resolution Protocol Daemon

rarpd program, 8–284

rexecd server, 8–299

protocol, 8–299

restricted, 8–300

RFC 822

address parsing by MH system, 8–10

date parsing, 8–78

ris program (RIS), 8–301

RK07 disk

formatting, 8–99

rlogind server

options, 8–303

rlogind server (cont.)
  protocol, 8–303, 8–303
  restricted, 8–303
RM0X disk
  formatting, 8–99
rmauth command, 8–116
rmt program
  commands, 8–305, 8–305
  restricted, 8–306
route program, 8–307
  See also routed program
  commands, 8–307
  diagnostics, 8–308
  restricted, 8–307
routed program, 8–309 to 8–311
  See also route program
  options, 8–311
  restricted, 8–311
routing table
  managing, 8–309 to 8–311
  manipulating, 8–307
RP0X disk
  formatting, 8–99
RPC
  displaying statistics, 8–248
RPC program number
  converting to DARPA protocol number, 8–270
RPC server
  reporting information, 8–312
rpcinfo command, 8–312
rrestore command, 8–313 to 8–316
  See also rmt program
  diagnostics, 8–315 to 8–316
  examples, 8–315
  keys, 8–313 to 8–315
rshd server, 8–317
  diagnostics, 8–318
  protocol, 8–317
  restricted, 8–318
  validating users, 8–317
RT-11 format device
  archiving, 8–13 to 8–17
ruptime command
  and the rwhod server, 8–320
rwalld daemon, 8–319
rwho command
  and the rwhod server, 8–320
rwhod server
  message format, 8–320, 8–320
RX02 floppy disk
  formatting, 8–322
rxformat program, 8–322
rzdisk utility
  options, 8–323, 8–323

S

sa command
  See also ac command
  options, 8–326, 8–326
savecore command, 8–328
SCSI disk
  managing, 8–323
secsetup command, 8–330
sendmail program, 8–331
setauth command, 8–116
setld
  subset fits on a system, 8–94
  verify software subsets, 8–109
setld utility, 8–337
shared memory
  exercising, 8–342
shell script
  mkconsole shell script, 8–211
shmx exerciser, 8–342
showmount command, 8–344
shutdown command, 8–345
Simple Network Management Protocol
  setting up automatically, 8–352
Simple Network Management Protocol Agent
  See also snmpd
sizer program, 8–347
SMP
  kgconv, 8–149
  startcpu, 8–353
  stopcpu, 8–357
snapcopy command, 8–349

snapshot file, 8–349
snmpd
  and /etc/snmpd.conf, 8–350
  debug levels defined, 8–351
  debugging, 8–351
  description, 8–350
  starting, 8–350
  trap support, 8–350
snmpsetup
  command, 8–352
  error messages, 8–352
software subset
  verifying, 8–109
software subsets
  installing and managing, 8–337
special file
  building, 8–214
start command (lpc), 8–189
statd
  network status daemon, 8–354
status command (lpc), 8–189
sticky file
  removing, 8–355, 8–356
stop command (lpc), 8–189
superblock
  updating, 8–392
svc.conf command
  options, 8–358
svcsetup
  command, 8–358
  restrictions, 8–358
swapon command, 8–360
syscript command
  See also cmx exerciser
  See also dskx exerciser
  See also fsx exerciser
  See also lpx exerciser
  See also memx exerciser
  See also mtx exerciser
  See also netx exerciser
  See also shmx exerciser, 8–361
syslog command, 8–362
  options, 8–363
  priority code list, 8–362
syslog command (cont.)
  restricted, 8–363
system
  backing up, 8–261
  logging message, 8–362
  printing statistics, 8–273
  rebooting, 8–291
  resetting date, 8–287
  resetting time, 8–287
  shutting down, 8–345
  sizing, 8–347
system crash
  error messages, 8–67, 8–69, 8–67, 8–69
system exerciser
  running, 8–361
system status server
  rwhod server, 8–320

T

talk program
  See also talkd server
talkd server, 8–365
tapex program, 8–366
  options, 8–366
  parameters, 8–369
TCP network
  testing, 8–238 to 8–239, 8–238e
TELNET protocol server
  See telnetd server
telnetd server, 8–372
terminal
  setting mode, 8–120
terminal communications system
  testing, 8–51
tftpd server, 8–373
timed
  daemon, 8–374
timedc, 8–377e
  options, 8–376
  timed control program, 8–376
topq printer command (lpc), 8–189
translation filters, 8–184

translator
Postscript (TM), 8–8
transliterate protocol trace
See trpt command
traps
authentication failure, 8–350
cold start, 8–350
trigger command, 8–378
Trivial File Transfer Protocol server
See tftpd server
trpt command, 8–379
tunefs command, 8–381
options, 8–381
restrictions, 8–382

U

uerf command, 8–383, 8–387
examples, 8–386, 8–391
options, 8–383, 8–387
restricted, 8–385
UFS
mounting, 8–226
ufs_mount command, 8–226
umount command (general), 8–220
umount command (NFS), 8–222
update program, 8–392
up-line load listener, 8–218
uuaids keyword, 8–393
uucico daemon, 8–395
uuclean program, 8–397
uucompact utility, 8–393
uucp file
setting up, 8–398
uucp utility
spool directories and, 8–393, 8–397
uucpsetup command, 8–398
uumkspool utility, 8–393
uumonitor program, 8–399
uupoll utility, 8–393
uurespool utility, 8–393

V

vipw command, 8–400

W

wtmp file
login accounting and, 8–4

X

Xerox NS Routing Information Protocol
See routed program

Y

yellow pages
svcorder file and, 8–411
YP binder, 8–409
YP data base lookup service
rebuilding data base, 8–404
YP environment
setting up, 8–411
YP map
copying to slave servers, 8–408
determining version, 8–407
transferring, 8–412
YP server, 8–409
ypbind daemon, 8–409
See also ypwhich command
ypmake command, 8–404
yppasswdd daemon, 8–406
yppoll command, 8–407
yppush command
See also ypxfr command, 8–408
ypserv command, 8–409
ypserv daemon
See also yppush command
ypsetup command, 8–411
ypxfr command, 8–412

How to Order Additional Documentation

Technical Support

If you need help deciding which documentation best meets your needs, call 800-343-4040 before placing your electronic, telephone, or direct mail order.

Electronic Orders

To place an order at the Electronic Store, dial 800-234-1998 using a 1200- or 2400-baud modem from anywhere in the USA, Canada, or Puerto Rico. If you need assistance using the Electronic Store, call 800-DIGITAL (800-344-4825).

Telephone and Direct Mail Orders

Your Location	Call	Contact
Continental USA, Alaska, or Hawaii	800-DIGITAL	Digital Equipment Corporation P.O. Box CS2008 Nashua, New Hampshire 03061
Puerto Rico	809-754-7575	Local Digital Subsidiary
Canada	800-267-6215	Digital Equipment of Canada Attn: DECdirect Operations KAO2/2 P.O. Box 13000 100 Herzberg Road Kanata, Ontario, Canada K2K 2A6
International	————	Local Digital subsidiary or approved distributor
Internal*	————	SSB Order Processing - WMO/E15 *or* Software Supply Business Digital Equipment Corporation Westminster, Massachusetts 01473

* For internal orders, you must submit an Internal Software Order Form (EN-01740-07).

Reader's Comments

ULTRIX
Reference Pages Section 8: Maintenance
AA-LY20B-TE

Please use this postage-paid form to comment on this manual. If you require a written reply to a software problem and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Thank you for your assistance.

Please rate this manual:	Excellent	Good	Fair	Poor
Accuracy (software works as manual says)	☐	☐	☐	☐
Completeness (enough information)	☐	☐	☐	☐
Clarity (easy to understand)	☐	☐	☐	☐
Organization (structure of subject matter)	☐	☐	☐	☐
Figures (useful)	☐	☐	☐	☐
Examples (useful)	☐	☐	☐	☐
Index (ability to find topic)	☐	☐	☐	☐
Page layout (easy to find information)	☐	☐	☐	☐

What would you like to see more/less of? ____________________

____

____

What do you like best about this manual? ____________________

____

____

What do you like least about this manual? ____________________

____

____

Please list errors you have found in this manual:

Page	Description
______	____________________
______	____________________
______	____________________
______	____________________
______	____________________

Additional comments or suggestions to improve this manual:

____

____

____

____

What version of the software described by this manual are you using? ____________

Name/Title ____________________ Dept. ____________

Company ____________________ Date ____________

Mailing Address ____________________

____________________ Email ____________________ Phone ____________

Do Not Tear – Fold Here and Tape

digital™

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST–CLASS MAIL PERMIT NO. 33 MAYNARD MA

POSTAGE WILL BE PAID BY ADDRESSEE

DIGITAL EQUIPMENT CORPORATION
OPEN SOFTWARE PUBLICATIONS MANAGER
ZKO3–2/Z04
110 SPIT BROOK ROAD
NASHUA NH 03062–9987

Do Not Tear – Fold Here

Cut
Along
Dotted
Line