IRIX User's Reference Manual

Volume III

Section 1, U–Z
Section 6

IRIS-4D Series

SiliconGraphics
Computer Systems

Document number: 007-0606-052

IRIX User's Reference Manual

Volume III

Version 5.2

Document Number 007-0606-052
(Includes *IRIX User's Reference Manual Version 5.0,*
Document Number 007-0606-050
and *IRIX User's Reference Manual Update Version 5.1,*
Document Number 007-0606-051)

IRIX User's Reference Manual, Volume III
Version 5.2
Document Number 007-0606-052
(Includes *IRIX User's Reference Manual Version 5.0,*
Document Number 007-0606-050
and *IRIX User's Reference Manual Update Version 5.1,*
Document Number 007-0606-051)

Silicon Graphics, Inc.
Mountain View, California

TABLE OF CONTENTS

NAME

intro – introduction to commands, application programs, and programming commands.

DESCRIPTION

This section describes, in alphabetical order, publicly-accessible commands. Certain distinctions of purpose are made in the headings:

(1)	Commands of general utility.
(1C)	Commands for communication with other systems.
(1G)	Graphics utilities.

Manual Page Command Syntax

Unless otherwise noted, commands described in the SYNOPSIS section of a manual page accept options and other arguments according to the following syntax and should be interpreted as explained below.

name [*–option* ...] [*cmdarg* ...]
where:

[ ]	Surround an *option* or *cmdarg* that is not required.
...	Indicates multiple occurrences of the *option* or *cmdarg*.
name	The name of an executable file.
option	This is either *noargletter*... or *argletter optarg*[,...] It is always preceded by a "–".

noargletter
A single letter representing an option without an option-argument. Note that more than one *noargletter* option can be grouped after one "–" (Rule 5, below).

argletter
A single letter representing an option requiring an option-argument.

optarg
An option-argument (character string) satisfying a preceding *argletter*. Note that groups of *optargs* following an *argletter* must be separated by commas, or separated by white space and quoted (Rule 8, below).

cmdarg
Path name (or other command argument) *not* beginning with "–", or "–" by itself indicating the standard input.

Command Syntax Standard: Rules

These command syntax rules are not followed by all current commands, but all new commands will obey them. *getopts*(1) should be used by all shell procedures to parse positional parameters and to check for legal options. It supports Rules 3-10 below. The enforcement of the other rules must be done by the command itself.

1. Command names (*name* above) must be between two and nine characters long.
2. Command names must include only lower-case letters and digits.
3. Option names (*option* above) must be one character long.
4. All options must be preceded by "–".
5. Options with no arguments may be grouped after a single "–".
6. The first option-argument (*optarg* above) following an option must be preceded by white space.
7. Option-arguments cannot be optional.
8. Groups of option-arguments following an option must either be separated by commas or separated by white space and quoted (e.g., `–o xxx,z,yy` or `–o "xxx z yy"`).
9. All options must precede operands (*cmdarg* above) on the command line.
10. "– –" may be used to indicate the end of the options.
11. The order of the options relative to one another should not matter.
12. The relative order of the operands (*cmdarg* above) may affect their significance in ways determined by the command with which they appear.
13. "–" preceded and followed by white space should only be used to mean standard input.

Throughout the manual pages there are references to *TMPDIR*, *BINDIR*, *INCDIR*, *LIBDIR*, and *LLIBDIR*. These represent directory names whose value is specified on each manual page as necessary. For example, *TMPDIR* might refer to /tmp or /usr/tmp. These are not environment variables and cannot be set. (There is also an environment variable called **TMPDIR** which can be set. See *tmpnam*(3S).)

SEE ALSO

getopts(1), exit(2), wait(2), getopt(3C).

DIAGNOSTICS

Upon termination, each command returns two bytes of status, one supplied by the system and giving the cause for termination, and (in the case of ''normal'' termination) one supplied by the program (see *wait*(2) and *exit*(2)). The former byte is 0 for normal termination; the latter is customarily 0 for successful execution and non-zero to indicate troubles such as erroneous parameters, or bad or inaccessible data. It is called variously ''exit code'', ''exit status'', or ''return code'', and is described only where special conventions are involved.

WARNINGS

Some commands produce unexpected results when processing files containing null characters. These commands often treat text input lines as strings and therefore become confused upon encountering a null character (the string terminator) within a line.

NAME

ul – do underlining

SYNOPSIS

ul [**–i**] [**–t** *terminal*] [*name* ...]

DESCRIPTION

ul reads the named files (or standard input if none are given) and translates occurrences of underscores to the sequence which indicates underlining for the terminal in use, as specified by the environment variable TERM. The –t option overrides the terminal kind specified in the environment. The file */usr/lib/terminfo* is read to determine the appropriate sequences for underlining. If the terminal is incapable of underlining, but is capable of a standout mode then that is used instead. If the terminal can overstrike, or handles underlining automatically, *ul* degenerates to *cat*(1). If the terminal cannot underline, underlining is ignored.

The –i option causes *ul* to indicate underlining onto by a separate line containing appropriate dashes '–'; this is useful when you want to look at the underlining which is present in an *nroff* output stream on a crt-terminal.

ul also handles bold printing, alternate character sets, etc. when the sequences are defined in the terminal's */usr/lib/terminfo* entry.

SEE ALSO

man(1), nroff(1)

AUTHOR

Mark Horton wrote *ul*. The –i option was originally a option of the editor *ex*(1), then an *iul* command.

BUGS

Nroff usually outputs a series of backspaces and underlines intermixed with the text to indicate underlining. No attempt is made to optimize the backward motion.

NAME

umask – set file-creation mode mask

SYNOPSIS

umask [ooo]

DESCRIPTION

The user file-creation mode mask is set to *ooo*. The three octal digits refer to read/write/execute permissions for *owner*, *group*, and *others*, respectively (see *chmod*(2) and *umask*(2)). The value of each specified digit is subtracted from the corresponding "digit" specified by the system for the creation of a file (see *creat*(2)). For example, **umask 022** removes *group* and *others* write permission (files normally created with mode **777** become mode **755**; files created with mode **666** become mode **644**).

If *ooo* is omitted, the current value of the mask is printed.

umask is recognized and executed by the shell.

umask can be included in the user's **.profile** (see *profile*(4)) and invoked at login to automatically set the user's permissions on files or directories created.

SEE ALSO

chmod(1), sh(1).
chmod(2), creat(2), umask(2), profile(4) in the *Programmer's Reference Manual.*

NAME

uname – identify the current IRIX system

SYNOPSIS

uname [**–snrvma**]

uname [–**S** system-name]

DESCRIPTION

uname prints information that identifies the current the IRIX system to standard output. The options cause selected information returned by *uname*(2) to be printed:

–s	Print the system name (the default).
–n	Print the nodename (the nodename is the name by which the system is known to a communications network).
–r	Print the operating system release. This string has one of the following forms: *m.n* or *m.n.a* where *m* is the major release number, *n* is the minor release number and *a* is the (optional) maintenance level of the release; e.g. **3.2** or **3.2.1**.
–v	Print the operating system version. This is the date and time that the operating system was generated, and has the form: *mmddhhmm*.
–m	Print the machine hardware name. This is the type of CPU board that the system is running on, e.g. **IP6**.
–a	Print all the above information.

The system name and the nodename may be changed by specifying a system name argument to the –**S** option. The system name argument is restricted to 8 characters. Only the super-user is allowed this capability.

SEE ALSO

uname(2) in the *Programmer's Reference Manual.*

NAME

unget – undo a previous get of an SCCS file

SYNOPSIS

unget [–r*SID*] [–**s**] [–**n**] files

DESCRIPTION

unget undoes the effect of a get –**e** done prior to creating the intended new delta. If a directory is named, *unget* behaves as though each file in the directory were specified as a named file, except that non-SCCS files and unreadable files are silently ignored. If a name of – is given, the standard input is read with each line being taken as the name of an SCCS file to be processed.

Keyletter arguments apply independently to each named file.

–**r***SID*	Uniquely identifies which delta is no longer intended. (This would have been specified by *get* as the "new delta"). The use of this keyletter is necessary only if two or more outstanding *get*s for editing on the same SCCS file were done by the same person (login name). A diagnostic results if the specified *SID* is ambiguous, or if it is necessary and omitted on the command line.
–**s**	Suppresses the printout, on the standard output, of the intended delta's *SID*.
–**n**	Causes the retention of the gotten file which would normally be removed from the current directory.

SEE ALSO

delta(1), get(1), sact(1).
help(1) in the *User's Reference Manual.*

DIAGNOSTICS

Use *help*(1) for explanations.

NAME

unifdef – strip or reduce ifdefs in C code

SYNOPSIS

unifdef [–**D***name*] [–**D***name*=*string*] [–**U***name*] [–**o***output*] [**-z**] [*input...*]

DESCRIPTION

Unifdef reads C source files and prints all input lines except those excluded by **#ifdef** constructs which refer to specified identifiers.

The –**D** option defines *name* as a macro with the value 1, causing *unifdef* to simplify

```
#ifdef name
Included text
#else
Excluded text
#endif
```

to just

```
Included text
```

and contrariwise for **#ifndef**.

The –**U** option works like –**D** except that *name* is undefined, so the **#ifdef** above would simplify to

```
Excluded text
```

-**D***name*=*string* causes *unifdef* to replace occurrences of *name* with *string* when evaluating **#if** and **#elif** expressions. For example, **unifdef –DBSD=43** would rewrite

```
#if BSD == 43
4.3BSD code
#elif BSD == 42
4.2BSD code
#endif
```

as

```
4.3BSD code
```

All **#ifdef** constructs which refer to names not defined with –**D** or undefined with –**U** are passed unchanged to output. *Unifdef* simplifies **#if** expressions which mix defined or undefined names with unknown names to express operations on just the unknown names. Thus **unifdef –Dsgi** would rewrite

```
#if defined sgi && !defined KERNEL
```

as

```
#ifndef KERNEL
```

and would preserve associated **#else** and **#endif** sections. *Unifdef* simplifies **#elif** chains as if they consisted of **#if** constructs nested within **#else** sections. For example, **unifdef –DBSD=43** would rewrite

```
#if SYSV
System V code
#elif BSD == 42
4.2BSD code
#elif BSD == 43
4.3BSD code
#elif XENIX
Xenix code
#endif
```

as

```
#if SYSV
System V code
#else
4.3BSD code
#if XENIX
Xenix code
#endif
#endif
```

Unifdef rewrites **#else** and **#endif** lines to conform to ANSI C, by enclosing any tokens after these keywords with comment delimiters. For example,

```
#ifdef DEBUG
Debugging code
#endif DEBUG
```

would be transcribed as

```
#ifdef DEBUG
Debugging code
#endif /* DEBUG */
```

Unifdef writes to standard output by default. The **–o** option may be used in lieu of shell redirection to specify an output file. More important, the file specified by *output* may be identical to one of the *input* files, or to standard input if there are no arguments. In this case, *unifdef* will safely overwrite the input file.

The −z option causes *unifdef* to simplify or eliminate sections controlled by **#if** constructs that test constant integer expressions. By default, constructs such as

```
#if 0
Excluded text
#endif
```

are left intact.

AUTHOR

Brendan Eich, 03/16/89

SEE ALSO

cc(1), cpp(1).

NAME

uniq – report repeated lines in a file

SYNOPSIS

uniq [**–udc** [+n] [–n]] [input [output]]

DESCRIPTION

uniq reads the input file comparing adjacent lines. In the normal case, the second and succeeding copies of repeated lines are removed; the remainder is written on the output file. *Input* and *output* should always be different. Note that repeated lines must be adjacent in order to be found; see *sort*(1). If the **–u** flag is used, just the lines that are not repeated in the original file are output. The **–d** option specifies that one copy of just the repeated lines is to be written. The normal mode output is the union of the **–u** and **–d** mode outputs.

The **–c** option supersedes **–u** and **–d** and generates an output report in default style but with each line preceded by a count of the number of times it occurred.

The *n* arguments specify skipping an initial portion of each line in the comparison:

–n The first *n* fields together with any blanks before each are ignored. A field is defined as a string of non-space, non-tab characters separated by tabs and spaces from its neighbors.

+n The first *n* characters are ignored. Fields are skipped before characters.

SEE ALSO

comm(1), sort(1).

NAME

units – conversion program

SYNOPSIS

units

DESCRIPTION

units converts quantities expressed in various standard scales to their equivalents in other scales. It works interactively in this fashion:

```
You have:  inch
You want:  cm
           * 2.540000e+00
           / 3.937008e-01
```

A quantity is specified as a multiplicative combination of units optionally preceded by a numeric multiplier. Powers are indicated by suffixed positive integers, division by the usual sign:

```
You have:  15 lbs force/in2
You want:  atm
           * 1.020689e+00
           / 9.797299e-01
```

units only does multiplicative scale changes; thus it can convert Kelvin to Rankine, but not Celsius to Fahrenheit. Most familiar units, abbreviations, and metric prefixes are recognized, together with a generous leavening of exotica and a few constants of nature including:

pi	ratio of circumference to diameter,
c	speed of light,
e	charge on an electron,
g	acceleration of gravity,
force	same as **g**,
mole	Avogadro's number,
water	pressure head per unit height of water,
au	astronomical unit.

Pound is not recognized as a unit of mass; **lb** is. Compound names are run together, (e.g., **lightyear**). British units that differ from their U.S. counterparts are prefixed thus: **brgallon**. For a complete list of units, type:

```
cat /usr/lib/unittab
```

FILES

/usr/lib/unittab

NAME

uptime – show how long system has been up

SYNOPSIS

uptime

DESCRIPTION

Uptime prints the current time, the length of time the system has been up, and the average number of jobs in the run queue over the last 1, 5 and 15 minutes. It is, essentially, the first line of a *w*(1) command.

FILES

/unix	system name list

SEE ALSO

w(1)

NAME

uucp, uulog, uuname – UNIX-to-UNIX system copy

SYNOPSIS

uucp [options] source-files destination-file
uulog [options] –ssystem
uulog [options] system
uulog [options] –fsystem
uuname [**–l**] [**–c**]

DESCRIPTION

uucp

uucp copies files named by the *source-file* arguments to the *destination-file* argument. A file name may be a path name on your machine, or may have the form:

system-name!path-name

where *system-name* is taken from a list of system names that *uucp* knows about. The *system-name* may also be a list of names such as

system-name!system-name!...!system-name!path-name

in which case an attempt is made to send the file via the specified route, to the destination. See WARNINGS and BUGS below for restrictions. Care should be taken to ensure that intermediate nodes in the route are willing to foward information (see WARNINGS below for restrictions).

The shell metacharacters ?, * and [...] appearing in *path-name* will be expanded on the appropriate system.

Path names may be one of:

(1) a full path name;

(2) a path name preceded by ˜*user* where *user* is a login name on the specified system and is replaced by that user's login directory;

(3) a path name preceded by ˜/*destination* where *destination* is appended to **/usr/spool/uucppublic**; (NOTE: This destination will be treated as a file name unless more than one file is being transfered by this request or the destination is already a directory. To ensure that it is a directory, follow the destination with a '/'. For example ˜/dan/ as the destination will make the directory /usr/spool/uucppublic/dan if it does not exist and put the requested file(s) in that directory).

(4) anything else is prefixed by the current directory.

If the result is an erroneous path name for the remote system the copy will fail. If the *destination-file* is a directory, the last part of the *source-file* name is used.

uucp preserves execute permissions across the transmission and gives 0666 read and write permissions (see *chmod*(2)).

The following options are interpreted by *uucp*:

–c	Do not copy local file to the spool directory for transfer to the remote machine (default).
–C	Force the copy of local files to the spool directory for transfer.
–d	Make all necessary directories for the file copy (default).
–f	Do not make intermediate directories for the file copy.
–g*grade*	*Grade* is a single letter/number; lower ascii sequence characters will cause the job to be transmitted earlier during a particular conversation.
–j	Output the job identification ASCII string on the standard output. This job identification can be used by *uustat* to obtain the status or terminate a job.
–m	Send mail to the requester when the copy is completed.
–n*user*	Notify *user* on the remote system that a file was sent.
–r	Do not start the file transfer, just queue the job.
–s*file*	Report status of the transfer to *file*. Note that the *file* must be a full path name.
–x*debug_level*	Produce debugging output on standard output. The *debug_level* is a number between 0 and 9; higher numbers give more detailed information. (Debugging will not be available if **uucp** was compiles with -DSMALL.)

uulog

uulog queries a log file of *uucp* or *uuxqt* transactions in a file **/usr/spool/uucp/.Log/uucico/***system*, or **/usr/spool/uucp/.Log/uuxqt/***system.*

The options cause *uulog* to print logging information:

–s*sys*	Print information about file transfer work involving system *sys*.

–f*system*	Does a "tail –f" of the file transfer log for *system*. (You must hit BREAK to exit this function.) Other options used in conjunction with the above:
–x	Look in the *uuxqt* log file for the given system.
–*number*	Indicates that a "tail" command of *number* lines should be executed.

uuname

uuname lists the names of systems known to *uucp*. The **–c** option returns the names of systems known to *cu*. (The two lists are the same, unless your machine is using different *Systems* files for *cu* and *uucp*. See the *Sysfiles* file.) The **–l** option returns the local system name.

FILES

/usr/spool/uucp	spool directories
/usr/spool/uucppublic/*	public directory for receiving and sending (**/usr/spool/uucppublic**)
/usr/lib/uucp/*	other data and program files

SEE ALSO

mail(1), uustat(1C), uucico(1M), uux(1C), uuxqt(1M).
chmod(2) in the *Programmer's Reference Manual.*

WARNINGS

The domain of remotely accessible files can (and for obvious security reasons, usually should) be severely restricted. You will very likely not be able to fetch files by path name; ask a responsible person on the remote system to send them to you. For the same reasons you will probably not be able to send files to arbitrary path names. As distributed, the remotely accessible files are those whose names begin **/usr/spool/uucppublic** (equivalent to ~/).

All files received by *uucp* will be owned by *uucp*.
The **–m** option will only work sending files or receiving a single file. Receiving multiple files specified by special shell characters **? * [...]** will not activate the **–m** option.

The forwarding of files through other systems may not be compatible with the previous version of *uucp*. If forwarding is used, all systems in the route must have the same version of *uucp*.

BUGS

Protected files and files that are in protected directories that are owned by the requestor can be sent by *uucp*. However, if the requestor is root, and the directory is not searchable by "other" or the file is not readable by "other", the request will fail.

NAME

uuencode, uudecode – encode/decode a binary file for transmission via mail

SYNOPSIS

uuencode [source] remotedest | **mail** sys1!sys2!..!decode
uudecode [file]

DESCRIPTION

Uuencode and *uudecode* are used to send a binary file via uucp (or other) mail. This combination can be used over indirect mail links even when *uusend*(1C) is not available.

Uuencode takes the named source file (default standard input) and produces an encoded version on the standard output. The encoding uses only printing ASCII characters, and includes the mode of the file and the *remotedest* for recreation on the remote system.

Uudecode reads an encoded file, strips off any leading and trailing lines added by mailers, and recreates the original file with the specified mode and name.

The intent is that all mail to the user "decode" should be filtered through the *uudecode* program. This way the file is created automatically without human intervention. This is possible on the uucp network by either using *sendmail* or by making *rmail* be a link to *Mail* instead of *mail*. In each case, an alias must be created in a master file to get the automatic invocation of *uudecode*.

If these facilities are not available, the file can be sent to a user on the remote machine who can uudecode it manually.

The encode file has an ordinary text form and can be edited by any text editor to change the mode or remote name.

SEE ALSO

uucp(1C), uux(1C), mail(1), uuencode(4)

BUGS

The file is expanded by 35% (3 bytes become 4 plus control information) causing it to take longer to transmit.

The user on the remote system who is invoking *uudecode* (often *uucp*) must have write permission on the specified file.

NAME

uustat – uucp status inquiry and job control

SYNOPSIS

uustat [–a]
uustat [–m]
uustat [–p]
uustat [–q]
uustat [–k*jobid*]
uustat [–r*jobid*]
uustat [–s*system*] [–u*user*]

DESCRIPTION

uustat will display the status of, or cancel, previously specified *uucp* commands, or provide general status on *uucp* connections to other systems. Only one of the following options can be specified with *uustat* per command execution:

–a	Output all jobs in queue.
–m	Report the status of accessibility of all machines.
–p	Execute a "ps –flp" for all the process-ids that are in the lock files.
–q	List the jobs queued for each machine. If a status file exists for the machine, its date, time and status information are reported. In addition, if a number appears in () next to the number of C or X files, it is the age in days of the oldest C./X. file for that system. The Retry field represents the number of hours until the next possible call. The Count is the number of failure attempts. NOTE: for systems with a moderate number of outstanding jobs, this could take 30 seconds or more of real-time to execute. As an example of the output produced by the –q option:

```
eagle      3C    04/07-11:07   NO DEVICES AVAILABLE
mh3bs3     2C    07/07-10:42   SUCCESSFUL
```

The above output tells how many command files are waiting for each system. Each command file may have zero or more files to be sent (zero means to call the system and see if work is to be done). The date and time refer to the previous interaction with the system followed by the status of the interaction.

–k*jobid* Kill the *uucp* request whose job identification is *jobid*. The killed *uucp* request must belong to the person issuing the *uustat* command unless one is the super-user.

−r*jobid* Rejuvenate *jobid*. The files associated with *jobid* are touched so that their modification time is set to the current time. This prevents the cleanup daemon from deleting the job until the jobs modification time reaches the limit imposed by the deamon.

Either or both of the following options can be specified with *uustat*:

−s*sys*	Report the status of all *uucp* requests for remote system *sys*.
−**u***user*	Report the status of all *uucp* requests issued by *user*.

Output for both the −s and −**u** options has the following format:

```
eaglen0000  4/07-11:01:03        (POLL)
eagleN1bd7  4/07-11:07           Seagledan522 /usr/dan/A
eagleC1bd8  4/07-11:07           Seagledan59 D.3b2al2ce4924
                  4/07-11:07     Seagledanrmail mike
```

With the above two options, the first field is the *jobid* of the job. This is followed by the date/time. The next field is either an 'S' or 'R' depending on whether the job is to send or request a file. This is followed by the user-id of the user who queued the job. The next field contains the size of the file, or in the case of a remote execution (*rmail* - the command used for remote mail), the name of the command. When the size appears in this field, the file name is also given. This can either be the name given by the user or an internal name (e.g., D.3b2alce4924) that is created for data files associated with remote executions (*rmail* in this example).
When no options are given, *uustat* outputs the status of all *uucp* requests issued by the current user.

FILES

/usr/spool/uucp/*	spool directories

SEE ALSO

uucp(1C).

NAME

uuto, uupick – public UNIX-to-UNIX system file copy

SYNOPSIS

uuto [options] source-files destination
uupick [**–s** system]

DESCRIPTION

uuto sends *source-files* to *destination*. *uuto* uses the *uucp*(1C) facility to send files, while it allows the local system to control the file access. A source-file name is a path name on your machine. Destination has the form:

system!*user*

where *system* is taken from a list of system names that *uucp* knows about (see *uuname*). *User* is the login name of someone on the specified system.

Two *options* are available:

–p	Copy the source file into the spool directory before transmission.
–m	Send mail to the sender when the copy is complete.

The files (or sub-trees if directories are specified) are sent to PUBDIR on *system*, where PUBDIR is a public directory defined in the *uucp* source. By default this directory is /usr/spool/uucppublic. Specifically the files are sent to

PUBDIR/receive/*user*/*mysystem*/files.

The destined recipient is notified by *mail*(1) of the arrival of files.

Uupick accepts or rejects the files transmitted to the user. Specifically, *uupick* searches PUBDIR for files destined for the user. For each entry (file or directory) found, the following message is printed on the standard output:

from *system*: [file *file-name*] [dir *dirname*] **?**

Uupick then reads a line from the standard input to determine the disposition of the file:

<new-line>	Go on to next entry.
d	Delete the entry.
m [*dir*]	Move the entry to named directory *dir*. If *dir* is not specified as a complete path name (in which $HOME is legitimate), a destination relative to the current directory is assumed. If no destination is given, the default is the current directory.
a [*dir*]	Same as **m** except moving all the files sent from *system*.

p	Print the content of the file.
q	Stop.
EOT (control-d)	Same as **q**.
!*command*	Escape to the shell to do *command*.
*	Print a command summary.

Uupick invoked with the **-s***system* option will only search the PUBDIR for files sent from *system*.

FILES

PUBDIR /usr/spool/uucppublic	public directory

SEE ALSO

mail(1), uucp(1C), uustat(1C), uux(1C).
uucleanup(1M) in the *System Administrator's Reference Manual.*

WARNINGS

In order to send files that begin with a dot (e.g., .profile) the files must by qualified with a dot. For example: .profile, .prof*, .profil? are correct; whereas *prof*, ?profile are incorrect.

NAME

uux – UNIX-to-UNIX system command execution

SYNOPSIS

uux [options] command-string

DESCRIPTION

uux will gather zero or more files from various systems, execute a command on a specified system and then send standard output to a file on a specified system.

NOTE: For security reasons, most installations limit the list of commands executable on behalf of an incoming request from *uux*, permitting only the receipt of mail (see *mail*(1)). (Remote execution permissions are defined in **/usr/lib/uucp/Permissions.)**

The *command-string* is made up of one or more arguments that look like a shell command line, except that the command and file names may be prefixed by *system-name*!. A null *system-name* is interpreted as the local system.

File names may be one of

(1) a full path name;

(2) a path name preceded by ~*xxx* where *xxx* is a login name on the specified system and is replaced by that user's login directory;

(3) anything else is prefixed by the current directory.

As an example, the command

```
uux "!diff usg!/usr/dan/file1 pwba!/a4/dan/file2 > !~/dan/file.diff"
```

will get the *file1* and *file2* files from the ''usg'' and ''pwba'' machines, execute a *diff*(1) command and put the results in *file.diff* in the local PUBDIR/dan/ directory.

Any special shell characters such as <>;| should be quoted either by quoting the entire *command-string*, or quoting the special characters as individual arguments.

uux will attempt to get all files to the execution system. For files that are output files, the file name must be escaped using parentheses. For example, the command

```
uux a!cut -f1 b!/usr/file \(c!/usr/file\)
```

gets /usr/file from system "b" and sends it to system "a", performs a *cut* command on that file and sends the result of the *cut* command to system "c".

uux will notify you if the requested command on the remote system was disallowed. This notification can be turned off by the **−n** option. The response comes by remote mail from the remote machine.

The following *options* are interpreted by *uux*:

−	The standard input to *uux* is made the standard input to the *command-string*.
−a*name*	Use *name* as the user identification replacing the initiator user-id. (Notification will be returned to the user.)
−b	Return whatever standard input was provided to the *uux* command if the exit status is non-zero.
−c	Do not copy local file to the spool directory for transfer to the remote machine (default).
−C	Force the copy of local files to the spool directory for transfer.
−g*grade*	*Grade* is a single letter/number; lower ASCII sequence characters will cause the job to be transmitted earlier during a particular conversation.
−j	Output the jobid ASCII string on the standard output which is the job identification. This job identification can be used by *uustat* to obtain the status or terminate a job.
−n	Do not notify the user if the command fails.
−p	Same as −: The standard input to *uux* is made the standard input to the *command-string*.
−r	Do not start the file transfer, just queue the job.
−s*file*	Report status of the transfer in *file*.
−x*debug_level*	Produce debugging output on the standard output. The *debug_level* is a number between 0 and 9; higher numbers give more detailed information.
−z	Send success notification to the user.

FILES

/usr/lib/uucp/spool	spool directories
/usr/lib/uucp/Permissions	remote execution permissions
/usr/lib/uucp/*	other data and programs

SEE ALSO

cut(1), mail(1), uucp(1C), uustat(1C).

WARNINGS

Only the first command of a shell pipeline may have a *system-name*!. All other commands are executed on the system of the first command.
The use of the shell metacharacter * will probably not do what you want it to do. The shell tokens << and >> are not implemented.

The execution of commands on remote systems takes place in an execution directory known to the *uucp* system. All files required for the execution will be put into this directory unless they already reside on that machine. Therefore, the simple file name (without path or machine reference) must be unique within the *uux* request. The following command will NOT work:

```
uux "a!diff b!/usr/dan/xyz c!/usr/dan/xyz > !xyz.diff"
```

but the command

```
uux "a!diff a!/usr/dan/xyz c!/usr/dan/xyz > !xyz.diff"
```

will work. (If *diff* is a permitted command.)

BUGS

Protected files and files that are in protected directories that are owned by the requestor can be sent in commands using *uux*. However, if the requestor is root, and the directory is not searchable by "other", the request will fail.

NAME

uwm - a window manager for X

SYNTAX

uwm [-display *display*] [-f *filename*]

DESCRIPTION

The *uwm* program is a window manager for X.

When *uwm* is invoked, it searches a predefined search path to locate any *uwm* startup files. If no startup files exist, *uwm* initializes its built-in default file.

If startup files exist in any of the following locations, it adds the variables to the default variables. In the case of contention, the variables in the last file found override previous specifications. Files in the *uwm* search path are:

/usr/lib/X11/uwm/system.uwmrc
$HOME/.uwmrc

To use only the settings defined in a single startup file, include the variables **resetbindings**, **resetmenus**, **resetvariables** at the top of that specific startup file.

OPTIONS

-f *filename*
Names an alternate file as a *uwm* startup file.

STARTUP FILE VARIABLES

Variables are typically entered first, at the top of the startup file. By convention, **resetbindings**, **resetmenus**, and **resetvariables** head the list.

autoselect/noautoselect
places the menu cursor in the first menu item. If unspecified, the menu cursor is placed in the menu header when the menu is displayed.

background=*color*
specifies the default background color for popup sizing windows, menus, and icons. The default is to use the WhitePixel for the current screen.

bordercolor=*color*
specifies the default border color for popup sizing windows, menus, and icons. The default is to use the BlackPixel for the current screen.

borderwidth=*pixels*
: specifies the default width in pixels for borders around popup sizing windows, menus, and icons. The default is 2.

delta=*pixels*
: indicates the number of pixels the cursor is moved before the action is interpreted by the window manager as a command. (Also refer to the **delta** mouse action.)

foreground=*color*
: specifies the default foreground color for popup sizing windows, menus, and icons. The default is to use the BlackPixel for the current screen.

freeze/nofreeze
: locks all other client applications out of the server during certain window manager tasks, such as move and resize.

grid/nogrid
: displays a finely-ruled grid to help you position an icon or window during resize or move operations.

hiconpad=*pixels*
: indicates the number of pixels to pad an icon horizontally. The default is five pixels.

hmenupad=*pixels*
: indicates the amount of space in pixels that each menu item is padded to the left and to the right of the text.

borderwidth=*pixels*
: indicates the width in pixels of the border surrounding icons.

iconfont=*fontname*
: names the font that is displayed within icons. Font names for a given server can be obtained using *xlsfonts(1).*

maxcolors=*number*
: limits the number of colors the window manager can use in a given invocation. If set to zero, or not specified, *uwm* assumes no limit to the number of colors it can take from the color map. **maxcolors** counts colors as they are included in the file.

mborderwidth=*pixels*
: indicates the width in pixels of the border surrounding menus.

normali/nonormali
: places icons created with **f.newiconify** within the root window, even if it is placed partially off the screen. With **nonormali** the icon is placed exactly where the

cursor leaves it.

normalw/nonormalw
places window created with **f.newiconify** within the root window, even if it is placed partially off the screen. With **nonormalw** the window is placed exactly where the cursor leaves it.

push=*number*
moves a window *number* pixels or 1/*number* times the size of the window, depending on whether **pushabsolute** or **pushrelative** is specified. Use this variable in conjunction with **f.pushup, f.pushdown, f.pushright**, or **f.pushleft**.

pushabsolute/pushrelative
pushabsolute indicates that the number entered with push is equivalent to pixels. When an f.push (left, right, up, or down) function is called, the window is moved exactly that number of pixels.

pushrelative indicates that the number entered with the push variable represents a relative number. When an f.push function is called, the window is invisibly divided into the number of parts you entered with the push variable, and the window is moved one part.

resetbindings, resetmenus, and **resetvariables**
resets all previous function bindings, menus, and variable entries, specified in any startup file in the *uwm* search path, including those in the default environment. By convention, these variables are entered first in the startup file.

resizefont=*fontname*
identifies the font of the indicator that displays dimensions in the corner of the window as you resize windows. See *xlsfonts(1)* for obtaining font names.

resizerelative/noresizerelative
indicates whether or not resize operations should be done relative to moving edge or edges. By default, the dynamic rectangle uses the actual pointer location to define the new size.

reverse/noreverse
defines the display as black characters on a white background for the window manager windows and icons.

viconpad=*pixels* indicates the number of pixels to pad an icon vertically. Default is five pixels.

vmenupad=*pixels*
indicates the amount of space in pixels that the menu is padded above and below the text.

volume=*number* increases or decreases the base level volume set by the *xset(1)* command. Enter an integer from 0 to 7, 7 being the loudest.

zap/nozap causes ghost lines to follow the window or icon from its previous default location to its new location during a move or resize operation.

BINDING SYNTAX

function=[*control key(s)*]:[*context*]:*mouse events*:" *menu name* "

Function and mouse events are required input. Menu name is required with the *f.menu* function definition only.

Function

f.beep emits a beep from the keyboard. Loudness is determined by the volume variable.

f.circledown causes the top window that is obscuring another window to drop to the bottom of the stack of windows.

f.circleup exposes the lowest window that is obscured by other windows.

f.continue releases the window server display action after you stop action with the **f.pause** function.

f.focus directs all keyboard input to the selected window. To reset the focus to all windows, invoke *f.focus* from the root window.

f.iconify when implemented from a window, this function converts the window to its respective icon. When implemented from an icon, f.iconify converts the icon to its respective window.

f.kill kills the client that created a window.

f.lower lowers a window that is obstructing a window below it.

f.menu invokes a menu. Enclose 'menu name' in quotes if it contains blank characters or parentheses.

f.menu=[*control key(s)*]:[*context*]:*mouse events*:" *menu name* "

f.move	moves a window or icon to a new location, which becomes the default location.
f.moveopaque	moves a window or icon to a new screen location. When using this function, the entire window or icon is moved to the new screen location. The grid effect is not used with this function.
f.newiconify	allows you to create a window or icon and then position the window or icon in a new default location on the screen.
f.pause	temporarily stops all display action. To release the screen and immediately update all windows, use the **f.continue** function.
f.pushdown	moves a window down. The distance of the push is determined by the push variables.
f.pushleft	moves a window to the left. The distance of the push is determined by the push variables.
f.pushright	moves a window to the right. The distance of the push is determined by the push variables.
f.pushup	moves a window up. The distance of the push is determined by the push variables.
f.raise	raises a window that is being obstructed by a window above it.
f.refresh	results in exposure events being sent to the window server clients for all unobscured or partially obscured windows. The windows will not refresh correctly if the exposure events are not handled properly.
f.resize	resizes an existing window. Note that some clients, notably editors, react unpredictably if you resize the window while the client is running.
f.restart	causes the window manager application to restart, retracing the *uwm* search path and initializing the variables it finds.

Control Keys

By default, the window manager uses meta as its control key. It can also use ctrl, shift, lock, or null (no control key). Control keys must be entered in lower case, and can be abbreviated as: c, l, m, s for ctrl, lock, meta, and shift, respectively.

You can bind one, two, or no control keys to a function. Use the bar (|) character to combine control keys.

Note that client applications other than the window manager may use pointer button and control key combinations. If the window manager has bound these combinations for its own use, the client application will never see the requested pointer input.

Context

The context refers to the screen location of the cursor when a command is initiated. When you include a context entry in a binding, the cursor must be in that context or the function will not be activated. The window manager recognizes the following four contexts: icon, window, root, (null).

The root context refers to the root, or background window, A (null) context is indicated when the context field is left blank, and allows a function to be invoked from any screen location. Combine contexts using the bar (|) character.

Mouse Buttons

Any of the following mouse buttons are accepted in lower case and can be abbreviated as l, m, or r, respectively: left, middle, right.

With the specific button, you must identify the action of that button. Mouse actions can be:

down	function occurs when the specified button is pressed down.
up	function occurs when the specified button is released.
delta	indicates that the mouse must be moved the number of pixels specified with the delta variable before the specified function is invoked. The mouse can be moved in any direction to satisfy the delta requirement.

MENU DEFINITION

After binding a set of function keys and a menu name to **f.menu**, you must define the menu to be invoked, using the following syntax:

```
menu = " menu name " {
"item name" : "action"
        .
        .
        .
}
```

Enter the menu name exactly the way it is entered with the **f.menu** function or the window manager will not recognize the link. If the menu name contains blank strings, tabs or parentheses, it must be quoted here and in the f.menu function entry. You can enter as many menu items as your screen is long. You cannot scroll within menus.

Any menu entry that contains quotes, special characters, parentheses, tabs, or strings of blanks must be enclosed in double quotes. Follow the item name by a colon (:).

Menu Action

Window manager functions

Any function previously described. E.g., **f.move** or **f.iconify**.

Shell commands

Begin with an exclamation point (!) and set to run in background. You cannot include a new line character within a shell command.

Text strings

Text strings are placed in the window server's cut buffer.

Strings starting with an up arrow (^) will have a new line character appended to the string after the up arrow (^) has been stripped from it.

Strings starting with a bar character (|) will be copied as is after the bar character (|) has been stripped.

Color Menus

Use the following syntax to add color to menus:

menu = "*menu name*" (*color1:color2:color3:color4*) {
"*item name*" : (*color5 :color6*) : " *action* "

.

.

.

}

color1	Foreground color of the header.
color2	Background color of the header.
color3	Foreground color of the highlighter, the horizontal band of color that moves with the cursor within the menu.
color4	Background color of the highlighter.
color5	Foreground color for the individual menu item.
color6	Background color for the individual menu item.

Color Defaults

Colors default to the colors of the root window under any of the following conditions:

1) If you run out of color map entries, either before or during an invocation of *uwm.*

2) If you specify a foreground or background color that does not exist in the RGB color database of the server (see */usr/lib/X11/rgb.txt* for a sample) both the foreground and background colors default to the root window colors.

3) If you omit a foreground or background color, both the foreground and background colors default to the root window colors.

4) If the total number of colors specified in the startup file exceeds the number specified in the *maxcolors* variable.

5) If you specify no colors in the startup file.

Customizing Icon Names

Icon names may be editted by placing the pointer inside the icon and typing. The Backspace, Rubout and Delete keys may used to remove a character from the end of a line and Control-U may be used to delete the whole name.

EXAMPLES

The following sample startup file shows the default window manager options:

```
# Global variables
#
resetbindings;resetvariables;resetmenus
autoselect
delta=25
freeze
grid
hiconpad=5
hmenupad=6
iconfont=oldeng
menufont=timrom12b
resizefont=9x15
viconpad=5
vmenupad=3
volume=7
#
# Mouse button/key maps
```

```
#
# FUNCTION   KEYS  CONTEXT  BUTTON    MENU(if any)
# ========   ====  =======  ======    ============
f.menu =     meta  :        :left down   :"WINDOW OPS"
f.menu =     meta  :        :middle down :"EXTENDED WINDOW OPS"
f.move =     meta  :wli  :right down
f.circleup = meta  :root :right down
#
# Menu specifications
#
menu = "WINDOW OPS" {
"(De)Iconify":   f.iconify
Move:            f.move
Resize:          f.resize
Lower:           f.lower
Raise:           f.raise
}

menu = "EXTENDED WINDOW OPS" {
Create Window:                  !"xterm &"
Iconify at New Position:   f.lowericonify
Focus Keyboard on Window:       f.focus
Freeze All Windows:             f.pause
Unfreeze All Windows:           f.continue
Circulate Windows Up:           f.circleup
Circulate Windows Down:         f.circledown
}
```

RESTRICTIONS

The color specifications have no effect on a monochrome system.

Some versions of lex have a hard-wired input buffer limit of 200 characters with no checking for overflow. A .uwmrc file containing lines longer than this limit will cause unpredictable behavior when used with a version of uwm built on such a system.

FILES

/usr/lib/X11/uwm/system.uwmrc
$HOME/.uwmrc

SEE ALSO

X(1), Xserver(1), xset(1), xlsfonts(1)

COPYRIGHT

AUTHOR

M. Gancarz, DEC Ultrix Engineering Group, Merrimack, New Hampshire, using some algorithms originally by Bob Scheifler, MIT Laboratory for Computer Science.

NAME

vadmin – interactive system administration tool

SYNOPSIS

vadmin [**–x** xpos] [**–y** ypos] [**–w** width] [**–h** height] [**–t** title] [**–d** helpfile] [**–s**] [tools-directory]

DESCRIPTION

The *vadmin* command is used to display a set of executable files contained in *tools-directory. vadmin* scans through the *tools-directory* and displays those executable files as a collection of selectable icons is a window. The default *tools-directory* used is */usr/lib/vadmin.*

Command line options are available to specify the window size, title and position as well as the help textfile (*vhelp*(1)). The command line options include:

–x *xpos*	right justify the window at *xpos*. The –x option, when used in conjunction with the **–y** option, define the position of the window's lower-left corner to the screen pixel location *xpos,ypos*.
–y *ypos*	aligns the bottom window edge at *ypos*. The –y option, when used in conjunction with the **–x** option, define the position of the window's lower-left corner to the screen pixel location *xpos,ypos*.
–w *width*	sets the window's width to *width*.
–h *height*	sets the window's height to *height*.
–t *title*	sets the window title to *title*.
–s	omits the "Invoke tools as" option. *vadmin* will bypass checking for uid and execute files as the current uid.
–d *helpfile*	uses the textfile *helpfile* as the text to be displayed when the help option is selected from the window. The *helpfile* is displayed by using *vhelp*(1). The default helpfile is */usr/lib/vadmin/vadmin.hlp.*

FILES

/usr/lib/vadmin
/usr/lib/vadmin/filetypes

SEE ALSO

vhelp(1)

NAME

val – validate SCCS file

SYNOPSIS

val –

val [–s] [–rSID] [–mname] [–ytype] files

DESCRIPTION

val determines if the specified *file* is an SCCS file meeting the characteristics specified by the optional argument list. Arguments to *val* may appear in any order. The arguments consist of keyletter arguments, which begin with a –, and named files.

val has a special argument, –, which causes reading of the standard input until an end-of-file condition is detected. Each line read is independently processed as if it were a command line argument list.

val generates diagnostic messages on the standard output for each command line and file processed, and also returns a single 8-bit code upon exit as described below.

The keyletter arguments are defined as follows. The effects of any keyletter argument apply independently to each named file on the command line.

–s	The presence of this argument silences the diagnostic message normally generated on the standard output for any error that is detected while processing each named file on a given command line.
–r*SID*	The argument value *SID* (*S*CCS *ID*entification String) is an SCCS delta number. A check is made to determine if the *SID* is ambiguous (e. g., **r**1 is ambiguous because it physically does not exist but implies 1.1, 1.2, etc., which may exist) or invalid (e. g., **r**1.0 or **r**1.1.0 are invalid because neither case can exist as a valid delta number). If the *SID* is valid and not ambiguous, a check is made to determine if it actually exists.
–mname	The argument value *name* is compared with the s-1SCCS %M% keyword in *file*.
–ytype	The argument value *type* is compared with the SCCS %Y% keyword in *file*.

The 8-bit code returned by *val* is a disjunction of the possible errors, i. e., can be interpreted as a bit string where (moving from left to right) set bits are interpreted as follows:

bit 0 = missing file argument;
bit 1 = unknown or duplicate keyletter argument;
bit 2 = corrupted SCCS file;
bit 3 = cannot open file or file not SCCS;
bit 4 = *SID* is invalid or ambiguous;
bit 5 = *SID* does not exist;
bit 6 = %Y%, **-y** mismatch;
bit 7 = %M%, **-m** mismatch;

Note that *val* can process two or more files on a given command line and in turn can process multiple command lines (when reading the standard input). In these cases an aggregate code is returned – a logical **OR** of the codes generated for each command line and file processed.

SEE ALSO

admin(1), delta(1), get(1), prs(1).
help(1) in the *User's Reference Manual.*

DIAGNOSTICS

Use *help*(1) for explanations.

BUGS

val can process up to 50 files on a single command line. Any number above 50 will produce a **core** dump.

NAME

vc – version control

SYNOPSIS

vc [–a] [–t] [–cchar] [–s] [keyword=value ... keyword=value]

DESCRIPTION

The *vc* command copies lines from the standard input to the standard output under control of its *arguments* and *control statements* encountered in the standard input. In the process of performing the copy operation, user declared *keywords* may be replaced by their string *value* when they appear in plain text and/or control statements.

The copying of lines from the standard input to the standard output is conditional, based on tests (in control statements) of keyword values specified in control statements or as *vc* command arguments.

A control statement is a single line beginning with a control character, except as modified by the **–t** keyletter (see below). The default control character is colon (:), except as modified by the **–c** keyletter (see below). Input lines beginning with a backslash (\) followed by a control character are not control lines and are copied to the standard output with the backslash removed. Lines beginning with a backslash followed by a non-control character are copied in their entirety.

A keyword is composed of 9 or less alphanumerics; the first must be alphabetic. A value is any ASCII string that can be created with *ed*(1); a numeric value is an unsigned string of digits. Keyword values may not contain blanks or tabs.

Replacement of keywords by values is done whenever a keyword surrounded by control characters is encountered on a version control statement. The **–a** keyletter (see below) forces replacement of keywords in *all* lines of text. An uninterpreted control character may be included in a value by preceding it with \. If a literal \ is desired, then it too must be preceded by \.

Keyletter Arguments

–a Forces replacement of keywords surrounded by control characters with their assigned value in *all* text lines and not just in *vc* statements.

–t All characters from the beginning of a line up to and including the first *tab* character are ignored for the purpose of detecting a control statement. If one is found, all characters up to and including the *tab* are discarded.

−cchar	Specifies a control character to be used in place of :.
−s	Silences warning messages (not error) that are normally printed on the diagnostic output.

Version Control Statements

:dcl keyword[, ..., keyword]

Used to declare keywords. All keywords must be declared.

:asg keyword=value

Used to assign values to keywords. An **asg** statement overrides the assignment for the corresponding keyword on the *vc* command line and all previous **asg**'s for that keyword. Keywords declared, but not assigned values have null values.

:if condition

•
•
•

:end

Used to skip lines of the standard input. If the condition is true all lines between the *if* statement and the matching *end* statement are copied to the standard output. If the condition is false, all intervening lines are discarded, including control statements. Note that intervening *if* statements and matching *end* statements are recognized solely for the purpose of maintaining the proper *if-end* matching.

The syntax of a condition is:

```
<cond>        ::= [ "not" ] <or>
<or>          ::= <and> | <and> "|" <or>
<and>         ::= <exp> | <exp> "&" <and>
<exp>         ::= "(" <or> ")" | <value> <op> <value>
<op>          ::= "=" | "!=" | "<" | ">"
<value>       ::= <arbitrary ASCII string> | <numeric string>
```

The available operators and their meanings are:

=	equal
!=	not equal
&	and
|	or
>	greater than
<	less than
( )	used for logical groupings
not	may only occur immediately after the *if*, and when present, inverts the value of the entire condition

The > and < operate only on unsigned integer values (e.g., : 012 > 12 is false). All other operators take strings as arguments (e.g., : 012 != 12 is true). The precedence of the operators (from highest to lowest) is:

= != > <	all of equal precedence
&	
|	

Parentheses may be used to alter the order of precedence.
Values must be separated from operators or parentheses by at least one blank or tab.

::text
: Used for keyword replacement on lines that are copied to the standard output. The two leading control characters are removed, and keywords surrounded by control characters in text are replaced by their value before the line is copied to the output file. This action is independent of the **–a** keyletter.

:on

:off
: Turn on or off keyword replacement on all lines.

:ctl char
: Change the control character to char.

:msg message
: Prints the given message on the diagnostic output.

:err message
: Prints the given message followed by:

  ERROR: err statement on line ... (915)

  on the diagnostic output. *vc* halts execution, and returns an exit code of 1.

SEE ALSO

ed(1), help(1) in the *User's Reference Manual.*

DIAGNOSTICS

Use *help*(1) for explanations.

EXIT CODES

0 – normal

1 – any error

NAME

vi, view, vedit – screen-oriented (visual) display editor based on ex

SYNOPSIS

vi [–**t** tag] [–**r** file] [–**L**] [–**wn**] [–**R**] [–x] [–C] [–**c** command] file ...
view [–**t** tag] [–**r** file] [–**L**] [–**wn**] [–**R**] [–x] [–C] [–**c** command] file ...
vedit [–**t** tag] [–**r** file] [–**L**] [–**wn**] [–**R**] [–x] [–C] [–**c** command] file ...

DESCRIPTION

vi (visual) is a display-oriented text editor based on an underlying line editor *ex*(1). It is possible to use the command mode of *ex* from within *vi* and vice-versa. The visual commands are described on this manual page; how to set options (like automatically numbering lines and automatically starting a new output line when you type carriage return) and all *ex*(1) line editor commands are described on the *ex*(1) manual page.

When using *vi*, changes you make to the file are reflected in what you see on your terminal screen. The position of the cursor on the screen indicates the position within the file.

Invocation Options

The following invocation options are interpreted by *vi* (previously documented options are discussed in the **NOTES** section at the end of this manual page):

–t *tag*	Edit the file containing the *tag* and position the editor at its definition.
–r *file*	Edit *file* after an editor or system crash. (Recovers the version of *file* that was in the buffer when the crash occurred.)
–**L**	List the name of all files saved as the result of an editor or system crash.
–w*n*	Set the default window size to *n*. This is useful when using the editor over a slow speed line.
–**R**	**Readonly** mode; the **readonly** flag is set, preventing accidental overwriting of the file.
–**x**	Encryption option; when used, *vi* simulates the **X** command of *ex*(1) and prompts the user for a key. This key is used to encrypt and decrypt text using the algorithm of *crypt*(1). The **X** command makes an educated guess to determine whether text read in is encrypted or not. The temporary buffer file is encrypted also, using a transformed version of the key typed in for the –x option.

See *crypt*(1). Also, see the **WARNING** section at the end of this manual page.

–C	Encryption option; same as the **–x** option, except that *vi* simulates the **C** command of *ex*(1). The **C** command is like the **X** command of *ex*(1), except that all text read in is assumed to have been encrypted.
–c *command*	Begin editing by executing the specified editor *command* (usually a search or positioning command).

The *file* argument indicates one or more files to be edited.

The *view* invocation is the same as *vi* except that the **readonly** flag is set.

The *vedit* invocation is intended for beginners. It is the same as *vi* except that the **report** flag is set to 1, the **showmode** and **novice** flags are set, and **magic** is turned off. These defaults make it easier to learn how to use *vi*.

vi Modes

Command	Normal and initial mode. Other modes return to command mode upon completion. **ESC** (escape) is used to cancel a partial command.
Input	Entered by setting any of the following options: **a A i I o O c C s S R .** Arbitrary text may then be entered. Input mode is normally terminated with **ESC** character, or, abnormally, with an interrupt.
Last line	Reading input for **: / ?** or **!**; terminate by typing a carriage return; an interrupt cancels termination.

COMMAND SUMMARY

In the descriptions, CR stands for carriage return and ESC stands for the escape key.

Sample commands

← ↓ ↑ →	arrow keys move the cursor
h j k l	same as arrow keys
i*text***ESC**	insert *text*
cw*new***ESC**	change word to *new*
ea*s***ESC**	pluralize word (end of word; append s; escape from input state)
x	delete a character
dw	delete a word
dd	delete a line
3dd	delete 3 lines
u	undo previous change
ZZ	exit *vi*, saving changes
:q!CR	quit, discarding changes
*/text***CR**	search for *text*
^U ^D	scroll up or down
:*cmd***CR**	any *ex* or *ed* command

Counts before vi commands

Numbers may be typed as a prefix to some commands. They are interpreted in one of these ways.

line/column number	**z G |**
scroll amount	**^D ^U**
repeat effect	most of the rest

Interrupting, canceling

ESC	end insert or incomplete cmd
DEL	(delete or rubout) interrupts

File manipulation

ZZ	if file modified, write and exit; otherwise, exit
:wCR	write back changes
:w ! CR	forced write, if permission originally not valid
:qCR	quit
:q ! CR	quit, discard changes
:e *name***CR**	edit file *name*
:e ! CR	reedit, discard changes
:e + *name***CR**	edit, starting at end
:e +*n***CR**	edit starting at line *n*
:e #CR	edit alternate file

:e ! #CR	edit alternate file, discard changes
:w *name***CR**	write file *name*
:w ! *name***CR**	overwrite file *name*
:shCR	run shell, then return
: ! *cmd***CR**	run *cmd*, then return
:nCR	edit next file in arglist
:n *args***CR**	specify new arglist
^G	show current file and line
:tag *tag***CR**	position cursor to *tag* (see *ctags*(1)), save position
:popCR	return to previous tag's position

In general, any *ex* or *ed* command (such as *substitute* or *global*) may be typed, preceded by a colon and followed by a carriage return.

Positioning within file

^F	forward screen
^B	backward screen
^D	scroll down half screen
^U	scroll up half screen
*n***G**	go to the beginning of the specified line (end default), where *n* is a line number
/pat	next line matching *pat*
?pat	previous line matching *pat*
n	repeat last / or ? command
N	reverse last / or ? command
/pat/+n	nth line after *pat*
?pat?−n	nth line before *pat*
]]	next section/function
[[	previous section/function
(	beginning of sentence
)	end of sentence
{	beginning of paragraph
}	end of paragraph
%	find matching ( ) { or }
^]	:tag command using word after the cursor as the tag
^T	return to previous tag's position (:pop command)

Adjusting the screen

^L	clear and redraw window
^R	clear and redraw window if ^L is → key
zCR	redraw screen with current line at top of window
z−CR	redraw screen with current line at bottom of window
z.CR	redraw screen with current line at center of window
*/pat/***z−CR**	move *pat* line to bottom of window

z*n***.CR**	use *n*-line window
^E	scroll window down 1 line
^Y	scroll window up 1 line

Marking and returning

\`\`	move cursor to previous context
´´	move cursor to first non-white space in line
m*x*	mark current position with the ASCII lower-case letter *x*
\`*x*	move cursor to mark *x*
´*x*	move cursor to first non-white space in line marked by *x*

Line positioning

H	top line on screen
L	last line on screen
M	middle line on screen
+	next line, at first non-white
–	previous line, at first non-white
CR	return, same as +
↓ or **j**	next line, same column
↑ or **k**	previous line, same column

Character positioning

^	first non white-space character
0	beginning of line
$	end of line
l or →	forward
h or ←	backward
^H	same as ← (backspace)
space	same as → (space bar)
f*x*	find next *x*
F*x*	find previous **x**
t*x*	move to character prior to next *x*
T*x*	move to character following previous *x*
;	repeat last **f F t** or **T**
,	repeat inverse of last **f F t** or **T**
n|	move to column *n*
%	find matching ({) or }

Words, sentences, paragraphs

w	forward a word
b	back a word
e	end of word
)	to next sentence
}	to next paragraph

(	back a sentence
{	back a paragraph
W	forward a blank-delimited word
B	back a blank-delimited word
E	end of a blank-delimited word

Corrections during insert

^H	erase last character (backspace)
^W	erase last word
erase	your erase character, same as **^H** (backspace)
kill	your kill character, erase this line of input
\	quotes your erase and kill characters
ESC	ends insertion, back to command mode
DEL	interrupt, terminates insert mode
^D	backtab one character; reset left margin of *autoindent*
^^D	caret (^) followed by control-d (^D); backtab to beginning of line; do not reset left margin of *autoindent*
0^D	backtab to beginning of line; reset left margin of *autoindent*
^T	insert *shiftwidth* spaces.
^V	quote non-printable character

Insert and replace

a	append after cursor
A	append at end of line
i	insert before cursor
I	insert before first non-blank
o	open line below
O	open above
r*x*	replace single char with *x*
R*text*ESC	replace characters

Operators

Operators are followed by a cursor motion, and affect all text that would have been moved over. For example, since **w** moves over a word, **dw** deletes the word that would be moved over. Double the operator, e.g., **dd** to affect whole lines.

d	delete
c	change
y	yank lines to buffer
<	left shift

>	right shift
!	filter through command

Miscellaneous Operations

C	change rest of line (**c$**)
D	delete rest of line (**d$**)
s	substitute chars (**cl**)
S	substitute lines (**cc**)
J	join lines
x	delete characters (**dl**)
X	delete characters before cursor (**dh**)
Y	yank lines (**yy**)

Yank and Put

Put inserts the text most recently deleted or yanked; however, if a buffer is named (using the ASCII lower-case letters **a** - **z**), the text in that buffer is put instead.

3yy	yank 3 lines
3yl	yank 3 characters
p	put back text after cursor
P	put back text before cursor
"*x***p**	put from buffer *x*
"*x***y**	yank to buffer *x*
"*x***d**	delete into buffer *x*

Undo, Redo, Retrieve

u	undo last change
U	restore current line
.	repeat last change
"*d***p**	retrieve *d*'th last delete

AUTHOR

vi and *ex* were developed by The University of California, Berkeley California, Computer Science Division, Department of Electrical Engineering and Computer Science.

FILES

/tmp	default directory where temporary work files are placed; it can be changed using the **directory** option (see the *ex(1)* **set** command)
/usr/lib/terminfo/?/*	compiled terminal description database
/usr/lib/.COREterm/?/*	subset of compiled terminal description database

NOTES

Two options, although they continue to be supported, have been replaced in the documentation by options that follow the Command Syntax Standard (see *intro*(1)). A **–r** option that is not followed with an option-argument has been replaced by **–L** and *+command* has been replaced by **–c** *command*.

SEE ALSO

ctags(1), ed(1), edit(1), ex(1).
User's Guide.
curses/terminfo chapter of the *Programmer's Guide*.

WARNINGS

The encryption options are provided with the Security Administration Utilities package, which is available only in the United States.

Tampering with entries in */usr/lib/.COREterm/?/** or */usr/lib/terminfo/?/** (for example, changing or removing an entry) can affect programs such as *vi*(1) that expect the entry to be present and correct. In particular, removing the "dumb" terminal may cause unexpected problems.

BUGS

In insert mode, software tabs using **^T** work only immediately after the *autoindent*.

Left and right shifts on intelligent terminals do not make use of insert and delete character operations in the terminal.

NAME

vmsprep – VMS tape preparation aid

SYNOPSIS

vmsprep [–] [name ...]

DESCRIPTION

Vmsprep traverses hierarchies of files and prepares them for transportation to VMS. Since ANSI stardard tapes (the VMS standard) do not allow hierarchy, this program provides a method of flattening the hierarchy onto a tape in such a way that it can be unpacked on VMS to recreate the same tree structure.

For reasons best not described here, *vmsprep* will attempt to exclude all RCS and SCCS archives by ignoring all files or directories named 'RCS' or 'SCCS', or files starting with 's.' or ending in ',v'.

The output of *vmsprep* is a pair of files vmsprep.namelist and UNPACK.COM. vmsprep.namelist is a list of files to be placed on the tape in the format required by *ansitape*. If the first argument is '–' instead of a file or directory name, vmsprep will instead send the namelist to standard output, and place UNPACK.COM in /tmp to avoid attempting to write in the current directory. All of the files except UNPACK.COM will be placed on the tape under cryptic names. UNPACK.COM is a VMS command script which will recreate all of the necessary directories and then move the cryptically named files to their proper place.

A typical sequence would be:

```
        vmsprep – tree1 tree2 file | ansitape cln trees –
```

Then on the VMS machine

```
        mount MFA0: trees
        copy MFA0:*.*.* *
        @UNPACK
```

FILES

vmsprep.namelist
UNPACK.COM

DIAGNOSTICS

A warning is reported if a file or directory name contains a character not permitted in VMS names. The offending character is replaced by 'Z' and *vmsprep* continues.

SEE ALSO

ansitape(1)

BUGS

Extra periods in file names may not be dealt with optimally.

All files and directories to be moved must be descendants of the current working directory. Absolute path names and paths containing ".." will produce unpredictable results.

Since vmsprep uses find(1) internally, it does not follow symbolic links.

The exclusion of RCS and SCCS files should be controlled by a command line flag.

Assumes VMS v4.0 or greater for long file names.

ORIGIN

4.3BSD

NAME

w – who is on and what they are doing

SYNOPSIS

w [**–fhls**] [user]

DESCRIPTION

W prints a summary of the current activity on the system, including what each user is doing. The heading line shows the current time of day, how long the system has been up, the number of users logged into the system, and the load averages. The load average numbers give the number of jobs in the run queue averaged over 1, 5 and 15 minutes.

The fields output are: the users login name, the name of the tty the user is on, the host from which the user is logged in, the time the user logged on, the time since the user last typed anything, the CPU time used by all processes and their children on that terminal, the CPU time used by the currently active processes, the name and arguments of the current process.

The **–h** flag suppresses the heading. The **–s** flag asks for a short form of output. In the short form, the tty is abbreviated, the login time and cpu times are left off, as are the arguments to commands. **–l** gives the long output, which is the default. The **–f** option suppresses the ''from'' field.

If a *user* name is included, the output will be restricted to that user.

FILES

/etc/utmp
/dev/kmem

SEE ALSO

who(1), ps(1)

AUTHOR

Mark Horton

BUGS

The notion of the ''current process'' is muddy. The current algorithm is ''the highest numbered process on the terminal that is not ignoring interrupts, or, if there is none, the highest numbered process on the terminal''. This fails, for example, in critical sections of programs like the shell and editor, or when faulty programs running in the background fork and fail to ignore interrupts. (In cases where no process can be found, *w* prints ''–''.)

The CPU time is only an estimate, in particular, if someone leaves a background process running after logging out, the person currently on that terminal is ''charged'' with the time.

Background processes are not shown, even though they account for much of the load on the system.

Sometimes processes, typically those in the background, are printed with null or garbaged arguments. In these cases, the name of the command is printed in parentheses.

W does not know about the new conventions for detection of background jobs. It will sometimes find a background job instead of the right one.

NAME

wait – await completion of process

SYNOPSIS

wait [*n*]

DESCRIPTION

Wait for your background process whose process id is *n* and report its termination status. If *n* is omitted, all your shell's currently active background processes are waited for and the return code will be zero.

The shell itself executes *wait*, without creating a new process.

SEE ALSO

sh(1).

CAVEAT

If you get the error message *cannot fork, too many processes*, try using the *wait*(1) command to clean up your background processes. If this doesn't help, the system process table is probably full or you have too many active foreground processes. (There is a limit to the number of process ids associated with your login, and to the number the system can keep track of.)

BUGS

Not all the processes of a 3- or more-stage pipeline are children of the shell, and thus cannot be waited for.

If *n* is not an active process id, all your shell's currently active background processes are waited for and the return code will be zero.

NAME

wall – write to all users

SYNOPSIS

/etc/wall [-g group] [msg_file]

DESCRIPTION

wall reads *msg_file* (or its standard input if none specified) until an end-of-file. It then sends this message to all currently logged-in users preceded by:

Broadcast Message from ...

If the *-g* option is used then only those users in the specified group are sent the message. The list of logged-on users is derived from the file **/etc/utmp**.

It is used to warn all users, typically prior to shutting down the system.

The sender must be super-user to override any protections the users may have invoked (see *mesg*(1)).

FILES

/dev/*
/etc/utmp

SEE ALSO

mesg(1), write(1), who(1).

DIAGNOSTICS

"Cannot send to ..." when the open on a user's tty file fails.

NAME

wc – word count

SYNOPSIS

wc [**–lwc**] [names]

DESCRIPTION

wc counts lines, words, and characters in the named files, or in the standard input if no *names* appear. It also keeps a total count for all named files. A word is a maximal string of characters delimited by spaces, tabs, or new-lines.

The options **l**, **w**, and **c** may be used in any combination to specify that a subset of lines, words, and characters are to be reported. The default is **–lwc**.

When *names* are specified on the command line, they will be printed along with the counts.

NAME

what – identify SCCS files

SYNOPSIS

what [–s] files

DESCRIPTION

what searches the given files for all occurrences of the pattern that *get*(1) substitutes for %Z% (this is @(#) at this printing) and prints out what follows until the first ", >, new-line, \, or null character. For example, if the C program in file **f.c** contains

```
char ident[] = " @(#)identification information ";
```

and **f.c** is compiled to yield **f.o** and **a.out**, then the command

```
what f.c f.o a.out
```

will print

```
f.c:
        identification information
f.o:
        identification information
a.out:
        identification information
```

what is intended to be used in conjunction with the command *get*(1), which automatically inserts identifying information, but it can also be used where the information is inserted manually. Only one option exists:

–s	Quit after finding the first occurrence of pattern in each file.

SEE ALSO

get(1).
help(1) in the *User's Reference Manual.*

DIAGNOSTICS

Exit status is 0 if any matches are found, otherwise 1. Use *help*(1) for explanations.

BUGS

It is possible that an unintended occurrence of the pattern @(#) could be found just by chance, but this causes no harm in nearly all cases.

NAME

whatis – describe what a command is

SYNOPSIS

whatis command ...

DESCRIPTION

whatis looks up a given command and gives the header line from the manual section. You can then run the *man*(1) command to get more information. If the line starts 'name(section) ...' you can do 'man section name' to get the documentation for it. Try 'whatis ed' and then you should do 'man 1 ed' to get the manual.

whatis is actually just the **–f** option to the *man*(1) command.

FILES

/usr/catman/whatis	Data base

SEE ALSO

apropos(1), man(1).

NAME

whereis – locate source, binary, and or manual for program

SYNOPSIS

whereis [**–sbm**] [**–u**] [**–SBM** dir ... **–f**] name ...

DESCRIPTION

Whereis locates source/binary and manuals sections for specified files. The supplied names are first stripped of leading pathname components and any (single) trailing extension of the form ".ext", e.g. ".c". Prefixes of "s." resulting from use of source code control are also dealt with. *Whereis* then attempts to locate the desired program in a list of standard places. If any of the **–b,** –s or **–m** flags are given then *whereis* searches only for binaries, sources or manual sections respectively (or any two thereof). The **–u** flag may be used to search for unusual entries. A file is said to be unusual if it does not have one entry of each requested type. Thus "whereis -m -u *" asks for those files in the current directory which have no documentation.

Finally, the **–B** , **–M** , and –S flags may be used to change or otherwise limit the places where *whereis* searches. The **–f** file flags is used to terminate the last such directory list and signal the start of file names.

EXAMPLE

The following finds all the files in /usr/bin which are not documented in /usr/man/u_man/man1 with source in /usr/src/cmd:

```
cd /usr/bin
whereis –u –M /usr/man/u_man/man1 –S /usr/src/cmd –f *
```

FILES

/bin, /etc, /lib
/usr/{bin, sbin, bsd, etc, games, demos, lib, lbin}
/usr/local/{bin, etc, lib}
/usr/bin/X11
/usr/man/u_man/{man1, man2, man3, man4, man5, man6}
/usr/man/a_man/{man1, man7}
/usr/src/cmd

BUGS

Since the program uses *chdir*(2) to run faster, pathnames given with the **–M** –S and **–B** must be full; i.e. they must begin with a "/".

NAME

which – locate a program file including aliases and path (*csh* only)

SYNOPSIS

which [–**a**] [–**f**] [*name*...]

DESCRIPTION

Which takes a list of names and looks for the files which would be executed had these names been given as commands. Each argument is expanded if it is aliased, and searched for along the user's path. Aliases are taken from the user's .cshrc file. The current value of path is used. The –**a** option reports all instances rather than just the first one. With the –**f** (fast) option, *which* ignores the .cshrc file.

FILES

~/.cshrc	source of aliases

DIAGNOSTICS

A diagnostic is given for names that are aliased to one or more words, or if an executable file with the argument name was not found in the path.

BUGS

Must be executed by a csh, since only csh's know about aliases.

NAME

who – who is on the system

SYNOPSIS

who [**–uTlHqpdbrtas**] [**–n** *x*] [file] [file]

who am i

who am I

DESCRIPTION

who can list the user's name, terminal line, login time, elapsed time since activity occurred on the line, and the process-ID of the command interpreter (shell) for each current UNIX system user. It examines the **/etc/utmp** file at login time to obtain its information. If *file* is given, that file (which must be in *utmp*(4) format) is examined. Usually, *file* will be **/etc/wtmp**, which contains a history of all the logins since the file was last created.

who with the **am i** or **am I** option identifies the invoking user.

The general format for output is:

name [state] line time [idle] [pid] [comment] [exit]

The *name*, *line*, and *time* information is produced by all options except **–q**; the *state* information is produced only by **–T**; the *idle* and *pid* information is produced only by **–u** and **–l**; and the *comment* and *exit* information is produced only by **–a**. The information produced for **–p**, **–d**, and **–r** is explained during the discussion of each option, below.

With options, *who* can list logins, logoffs, reboots, and changes to the system clock, as well as other processes spawned by the *init* process. These options are:

–u This option lists only those users who are currently logged in. The *name* is the user's login name. The *line* is the name of the line as found in the directory **/dev**. The *time* is the time that the user logged in. The *idle* column contains the number of hours and minutes since activity last occurred on that particular line. A dot (**.**) indicates that the terminal has seen activity in the last minute and is therefore "current". If more than twenty-four hours have elapsed or the line has not been used since boot time, the entry is marked **old**. This field is useful when trying to determine whether a person is working at the terminal or not. The *pid* is the process-ID of the user's shell. The *comment* is the comment field associated with this line as found in **/etc/inittab** (see *inittab*(4)). This can contain information about where the terminal is located, the telephone number of the dataset, type of terminal if hard-wired, etc.

−T This option is the same as the **−s** option, except that the *state* of the terminal line is printed. The *state* describes whether someone else can write to that terminal. A + appears if the terminal is writable by anyone; a − appears if it is not. **root** can write to all lines having a + or a − in the *state* field. If a bad line is encountered, a ? is printed.

−l This option lists only those lines on which the system is waiting for someone to login. The *name* field is **LOGIN** in such cases. Other fields are the same as for user entries except that the *state* field does not exist.

−H This option will print column headings above the regular output.

−q This is a quick *who*, displaying only the names and the number of users currently logged on. When this option is used, all other options are ignored.

−p This option lists any other process which is currently active and has been previously spawned by *init*. The *name* field is the name of the program executed by *init* as found in **/etc/inittab**. The *state*, *line*, and *idle* fields have no meaning. The *comment* field shows the *id* field of the line from **/etc/inittab** that spawned this process. See *inittab*(4).

−d This option displays all processes that have expired and not been respawned by *init*. The *exit* field appears for dead processes and contains the termination and exit values (as returned by *wait*(2)), of the dead process. This can be useful in determining why a process terminated.

−b This option indicates the time and date of the last reboot.

−r This option indicates the current *run-level* of the *init* process. In addition, it produces the process termination status, process id, and process exit status (see *utmp*(4)) under the *idle*, *pid*, and *comment* headings, respectively.

−t This option indicates the last change to the system clock (via the *date*(1) command) by **root**. See *su*(1).

−a This option processes **/etc/utmp** or the named *file* with all options turned on.

−s This option is the default and lists only the *name*, *line*, and *time* fields.

−n *x* This option takes a numeric argument, *x*, which specifies the number of users to display per line. *x* must be at least **1**. The −n option must be used with −q.

Note to the super-user: after a shutdown to the single-user state, *who* returns a prompt; the reason is that since **/etc/utmp** is updated at login time and there is no login in single-user state, *who* cannot report accurately on this state. *who am i*, however, returns the correct information.

FILES

/etc/utmp
/etc/wtmp
/etc/inittab

SEE ALSO

date(1), login(1), mesg(1), su(1M).
init(1M) in the *System Administrator's Reference Manual.*
wait(2), inittab(4), utmp(4) in the *Programmer's Reference Manual.*

NAME

whoami – print effective current user id

SYNOPSIS

whoami

DESCRIPTION

Whoami prints who you are. It works even if you are su'd, while 'who am i' does not since it uses /etc/utmp.

FILES

/etc/passwd	Name data base

SEE ALSO

who (1)

NAME

winterm – utility to launch applications that require a terminal emulator.

SYNOPSIS

winterm [**–H**|**–f** font|**–t** title|**–p** x,y|**–s** cols,lines|**–c** command]

DESCRIPTION

winterm is a shell script that presents an abstract command line syntax for the user's own terminal emulator. Terminal emulators supported include *wsh*, *psterm*, and *xterm*. The user can preset their preferred termulator (with preferred options) by setting the environment variable *$WINTERM*. If *WINTERM* is unset, winterm provides *wsh* as a default.

–H	holds the winterm open.
–f font	sets the font used by the winterm to *font*.
–t title	sets the title used by the winterm to *title*
–p x,y	sets the position of lower left corner of the winterm to *x,y*.
–s cols,lines	sets the size of the winterm to *cols,lines*.
–c command	feeds the rest of the line as the command to execute. Must be the last flag set when winterm is invoked.

NOTES

The default *WINTERM* is: `WINTERM='wsh -fScreen11 -C54,96,3,2,0,50'`.
The –**H** option is not supported by *xterm*.
The –**f** option is not supported by *psterm*.

SEE ALSO

Programming the IRIS WorkSpace

NAME

WorkSpace – graphical interface to file system

SYNOPSIS

workspace [–w] [–**t** [name]]

DESCRIPTION

WorkSpace provides a graphical, interactive interface to the IRIX file system. This interface is provided via two kinds of views. When invoked with no arguments, WorkSpace opens a window displaying a portion of the IRIX file tree, which can be pruned and expanded on a per user basis. The second type of view, instantiated by opening a directory icon, provides an constantly up-to-date representation of that
IRIX directory.

WorkSpace accepts the following options.

–w Open only the WorkSpace view (Note that otherwise WorkSpace will start up with whatever views were open the last time it was used).

–**t** [name]
Allow the user to create a sample WorkSpace view, and install it in /usr/lib/workspace. If a *name* is provided after this option, the file will be installed in the form *name*.wsrc. Otherwise, the file will be called *$USER*.wsrc. Each time a new template is added, or one is altered, all users on the system will automatically load it the next time they start their WorkSpace. Note that the template file will have to be installed manually if this option is run by a user without privilege to write into */usr/lib/workspace.*

Note that only one WorkSpace process can be run for each user. If the process is invoked again while it is already running, the WorkSpace window will be either opened or popped to the top.

SEE ALSO

dirview(1G)

Programming the IRIS WorkSpace

Author

Betsy Zeller

NAME

write – write to another user

SYNOPSIS

write user [line]

DESCRIPTION

write copies lines from your terminal to that of another user. When first called, it sends the message:

Message from *yourname* **(tty??)** [*date*]...

to the person you want to talk to. When it has successfully completed the connection, it also sends two bells to your own terminal to indicate that what you are typing is being sent.

The recipient of the message should write back at this point. Communication continues until an end of file is read from the terminal, an interrupt is sent, or the recipient has executed "mesg n". At that point *write* writes **EOT** on the other terminal and exits.

If you want to write to a user who is logged in more than once, the *line* argument may be used to indicate which line or terminal to send to (e.g., **tty00**); otherwise, the first writable instance of the user found in **/etc/utmp** is assumed and the following message posted:

user is logged on more than one place.
You are connected to "*terminal*".
Other locations are:
terminal

Permission to write may be denied or granted by use of the *mesg*(1) command. Writing to others is normally allowed by default. Certain commands, such as *pr*(1) disallow messages in order to prevent interference with their output. However, if the user has super-user permissions, messages can be forced onto a write-inhibited terminal.

If the character **!** is found at the beginning of a line, *write* calls the shell to execute the rest of the line as a command.

The following protocol is suggested for using *write*: when you first *write* to another user, wait for them to *write* back before starting to send. Each person should end a message with a distinctive signal (i.e., **(o)** for "over") so that the other person knows when to reply. The signal **(oo)** (for "over and out") is suggested when conversation is to be terminated.

FILES

/etc/utmp	to find user
/bin/sh	to execute !

SEE ALSO

mail(1), mesg(1), pr(1), sh(1), who(1).

DIAGNOSTICS

``*user is not logged on*'' if the person you are trying to *write* to is not logged on.

``*Permission denied*'' if the person you are trying to *write* to denies that permission (with *mesg*).

``*Warning: cannot respond, set mesg -y*'' if your terminal is set to *mesg n* and the recipient cannot respond to you.

``*Can no longer write to user*'' if the recipient has denied permission (*mesg n*) after you had started writing.

NAME

wsh – creates and specifies a window shell

SYNOPSIS

wsh [**–C** textcolor,pagecolor,hilitecolor,cursorcolor]
[**–C** textcolor,pagecolor,hilitecolor,cursorcolor,selfg,selbg] [**–d**]
[**–E**] [**–f** font] [**–F**] [**–H**] [**–l** logfile] [**–L** dev1,dev2]
[**–m** cols,lines] [**–n** name] [**–p** x,y] [**–r** lines] [**–R** device]
[**–s** cols,lines] [**–t** title] [**–v**] [**–Z** number]
[**–c** cmd [args]]

DESCRIPTION

wsh is a terminal emulation program that runs a shell (or other UNIX command) within its own window on the screen. Each *wsh* provides menus that allow you to interactively attach and select windows, change fonts and window sizes, and create new *wsh*s.

Command line options are available to specify the font, window size, title, and position when *wsh* starts up. The command line options include:

–C *textcolor,pagecolor,hilitecolor,cursorcolor*
: Set the color map indices used by *wsh* to display text characters, the background page, highlighted text (reverse video) characters, and the block cursor.

–C *textcolor,pagecolor,hilitecolor,cursorcolor,selfg,selbg*
: Set the color map indices used by *wsh* to display text characters, the background page, highlighted text (reverse video) characters, the block cursor, the selection text characters, and the selection background.

–d Make *wsh* run in the foreground. This is for applications that use *wsh* as a subprocess, and need to know when *wsh* exits. Identical to the **-Z1** flag.

–E Inform *wsh* that it should hold the display when the command exits, if and only if the command exits with an error.

–f *font* Set the specified font for displaying text. For example, the following command line opens a *wsh* window using a 14-point Courier font:

```
wsh -f Courier14
```

–F Start *wsh* in fixed terminal size mode. The terminal size is fixed initially at the maximum window size specified by the **–m** command line option, or the default maximum size (80,40). While in fixed terminal size mode, making the window smaller will hide some of the fixed sized terminal's data. The "set size" menu may be used to set up a new fixed terminal size. By default, *wsh* starts up in variable terminal size mode. While in variable terminal size mode, reshaping the window causes the terminal itself to be resized accordingly. In this case, the terminal's data is never hidden because it is wrapped within the confines of the window itself.

−H Hold *wsh* after its child program has exited, to permit viewing its output. See −E for a more useful variation.

−l *logfile* Have *wsh* log the output of the child program into *logfile*.

−L *dev1,dev2*
Have *wsh* use a specific pty device, instead of allocating one. dev1 is the name of the pty master, dev2 is the name of the pty slave. This is normally used for the console window only.

−m *cols,lines*
Set the maximum window size, as the number of columns wide by the number of lines high. The window manager will not allow the window to be reshaped larger than the maximum window size. If the **−m** option is not given, the maximum window size is set to the initial window size, as specified by the **−s** command line option, or the default size. If the **−F** option is given, the maximum window size is also used as the initial fixed terminal size at startup.

−n *name* Provide a unique window name for a *wsh*. The window manager will (optionally) look up *name* in *user.ps* to find initial positioning information. If **−n** is given with no argument, the system uses the name of the command running within *wsh*. If no **−n** is given, the system does not ask *user.ps* for positioning information.

−p *x,y* Set the position of the lower-left corner of the window to the screen pixel location *x,y*. If the **−p** option (or *user.ps*, see **−n** above) does not provide a window location, the window manager prompts for it.

−r *lines* set the number of *lines* of text retained by *wsh*. Using the scrollbar one can view all of the lines retained by *wsh*. New lines entered by *wsh* are retained up to the limit specified by *lines;* after this, lines are deleted from the ``top'' of the list of lines. Setting *lines* to zero eliminates the scroll bar.

−R *device*
Provide an alternate (redirect) output device. When *wsh* receives a "toggle-redirect" command from a locally bound function key, the output of the *wsh* is redirected to the given device.

–s *cols,lines*
: Set the initial size of the window by specifying the number of columns wide by the number of lines high. The default size is 80,40. If the –**s** size given is larger than the default maximum window size (80,40), the effective maximum size expands to accommodate it.

–t *title* Set the title of the window to the *title*. If a null string is provided, (–**t** ""), *wsh* will appear without a title bar.

–v Make wsh run as a vt100 terminal emulator. Normally, wsh emulates the "iris-ansi" terminal, which is almost a vt100. This switch makes a few minor changes to the emulation for vt100 compatibility.

–Z1 Make *wsh* run in the foreground. This is for applications that use *wsh* as a subprocess, and need to know when *wsh* exits. This is identical to the –**d** flag.

–Z2 Tell *wsh* that it is going to run the console and to not go into the background.

–Z3 Tell *wsh* that it is going to run the console, not to go into the background, and to fork a shell.

–Z4 Normally *wsh* will change the process group of the child process (and the tty) during startup. This flag disables that behavior.

–Z5 Draw the *wsh* window without a 4Sight window frame. if you use this option, you cannot use the window manager to manipulate the window once you initially position it on the screen.

–Z6 Run *wsh* without a keyboard. Any data received from the main keyboard is ignored. Use this when you need an output only *wsh*.

–Z7 Start up *wsh* in an iconic form.

–c *cmd* [*args*]
: Execute a child program within the *wsh* window, using the specified command line arguments, rather than the default shell. The –**c** option will pass all trailing arguments to *wsh* , to be executed as a command line. Thus, if –**c** is given, it must appear as the final *wsh* option.

You can restore the size of a *wsh* window using the ``previous'' item on the window menu. This allows you to toggle back and forth between two window sizes and positions. Note that font changes which affect window size reset the previous size.

When you log on to a remote system, use vt100 if no iris-ansi entry exists in the terminfo or termcap file. You must set *wsh* terminal size to 80,24 for this to work properly. Note that certain vt100 capabilities are not present in *wsh* and thus certain foreign applications may not function properly.

When you use *vi, emacs,* or other visual-mode programs remotely, you must size the terminal window to match the terminal size referenced by the current $TERM environment variable.

wsh recognizes escape sequences as listed in the *4Sight Programmer's Guide,* Section 1, Appendix A, ''Terminal and Keyboard Data''.

When in alternate keypad mode (ESC =), the special function keys F9-F12 on the IRIS keyboard emulate the keys PF1-PF4 on a VT100 keyboard. (ESC P, ESC Q, ESC R, ESC S).

MENU USAGE

The *wsh* menu provides the following facilities:

select	Pop this *wsh* to the top of the window hierarchy.
copy	Copy selected text to the cut buffer. This text can be sent to a *wsh* window or to another *jot* window, as well as the window it was copied from.
send	Send the data in the cut buffer to the child program, as if it had been typed. Clicking the middle mouse will send any copied text from the cut buffer to the child process.
size	Size provides a submenu which allows control over the size of terminal that *wsh* is emulating.
font	Font provides a submenu which leads to further submenus, all of which all you to pick a new font and font size to use.
clone	Clone makes a visual duplicate of *wsh*. The duplicate will have the same size, color, and font choice as its progenitor. It will run the same command if the **–c** argument is given.

ADVANCED FEATURES

wsh now provides a text selection facility that allows text to be transferred from the display of one *wsh* window and sent as input to any other *wsh* window.

Select text with the left mouse button by performing the following actions:

click-hold-release	This action selects all text between the the point where the left mouse button is first clicked down and the point where it is released.

click-click	Double-clicking the left mouse causes the word over which the mouse cursor is clicked to be selected.
shift&click-hold-release	If you have already selected a block of text, you can extend your selection either forwards or backwards by holding down the shift key while clicking the left mouse.
alt&click-hold-release	Same as *click-hold-release*, except that an implicit copy (to the cut buffer) is performed when the left mouse button is released. You can use *alt&shift&click* as well.

Clicking the middle mouse will send any copied text from the cut buffer to the child process.

SCROLL BAR

wsh now has an improved scroll bar with a proportional thumb.

Clicking on the up and down arrows with the left mouse button causes the *wsh* window to scroll up or down one line, respectively. Holding down the left button over either arrow causes the scrolling to auto-repeat. Holding down the shift key while clicking on either arrow causes the window to scroll up or down one full page of text.

The scroll bar's thumb can also be used to scroll through the *wsh* window by clicking and holding down the left mouse button while dragging the mouse cursor up and down. The size of the thumb is variable, and indicates the percentage of total text stored in the window's buffer that is currently visible in the window.

KEY BINDING

wsh supports limited key binding through the *bindkey*(1) command. See the *bindkey* man page for details.

FILES

user.ps
/usr/lib/fmfonts

SEE ALSO

bindkey(1), jot(1G)
4Sight Programmer's Guide, Section 1, Chapter 5, "Using *wsh* with GL Clients", and Appendix A, "Terminal and Keyboard Data".

NAME

X - a portable, network-transparent window system

SYNOPSIS

The X Window System is a network transparent window system developed at MIT which runs on a wide range of computing and graphics machines. The core distribution from MIT has support for the following operating systems:

Ultrix 3.1 (Digital)
SunOS 4.0.3 (Sun)
HP-UX 6.5 (Hewlett-Packard)
Domain/OS 10.1 (HP/Apollo)
A/UX 1.1 (Apple)
AIX RT-2.2 and PS/2-1.1 (IBM)
AOS-4.3 (IBM)
UTEK 4.0 (Tektronix)
NEWS-OS 3.2 (Sony; client only)
UNICOS 5.0.1 (Cray; client only)
UNIX(tm) System V, Release 3.2 (AT&T 6386 WGS; client only)

It should be relatively easy to build the client-side software on a variety of other system. Commercial implementations are also available for a wide range of platforms.

The X Consortium requests that the following names be used when referring to this software:

X
X Window System
X Version 11
X Window System, Version 11
X11

X Window System is a trademark of the Massachusetts Institute of Technology.

DESCRIPTION

X Window System servers run on computers with bitmap displays. The server distributes user input to and accepts output requests from various client programs through a variety of different interprocess communication channels. Although the most common case is for the client programs to be running on the same machine as the server, clients can be run transparently from other machines (including machines with different architectures and operating systems) as well.

X supports overlapping hierarchical subwindows and text and graphics operations, on both monochrome and color displays. For a full explanation of the functions that are available, see the *Xlib - C Language X Interface* manual, the *X Window System Protocol* specification, the *X Toolkit Intrinsics - C Language Interface* manual, and various toolkit documents.

The number of programs that use *X* is growing rapidly. Of particular interest are: a terminal emulator (*xterm*), a window manager (*twm*), a display manager (*xdm*), mail managing utilities (*xmh* and *xbiff*), a manual page browser (*xman*), a bitmap editor (*bitmap*), access control programs (*xauth* and *xhost*), user preference setting programs (*xrdb*, *xset*, *xsetroot*, and *xmodmap*), a load monitor (*xload*), clocks (*xclock* and *oclock*), a font displayer (*xfd*), utilities for listing information about fonts, windows, and displays (*xlsfonts*, *xfontsel*, *xlswins*, *xwininfo*, *xlsclients*, *xdpyinfo*, and *xprop*), a diagnostic for seeing what events are generated and when (*xev*), screen image manipulation utilities (*xwd*, *xwud*, *xpr*, and *xmag*), and various demos (*xeyes*, *ico*, *muncher*, *puzzle*, *xgc*, etc.).

Many other utilities, window managers, games, toolkits, etc. are available from the user-contributed software. See your site administrator for details.

STARTING UP

There are two main ways of getting the X server and an initial set of client applications started. The particular method used depends on what operating system you are running and on whether or not you use other window systems in addition to X.

xdm (the X Display Manager)

If you want to always have X running on your display, your site administrator can set your machine up to use the X Display Manager *xdm*. This program is typically started by the system at boot time and takes care of keeping the server running and getting users logged in. If you are running *xdm*, you will see a window on the screen welcoming you to the system and asking for your username and password. Simply type them in as you would at a normal terminal, pressing the Return key after each. If you make a mistake, *xdm* will display an error message and ask you to try again. After you have successfully logged in, *xdm* will start up your X environment. By default, if you have an executable file named *.xsession* in your home directory, *xdm* will treat it as a program (or shell script) to run to start up your initial clients (such as terminal emulators, clocks, a window manager, user settings for things like the background, the speed of the pointer, etc.). Your site administrator can provide details.

xinit **(run manually from the shell)**

Sites that support more than one window system might choose to use the *xinit* program for starting X manually. If this is true for your machine, your site administrator will probably have provided a program named "x11", "startx", or "xstart" that will do site-specific initialization (such as loading convenient default resources, running a window manager, displaying a clock, and starting several terminal emulators) in a nice way. If not, you can build such a script using the *xinit* program. This utility simply runs one user-specified program to start the server, runs another to start up any desired clients, and then waits for either to finish. Since either or both of the user-specified programs may be a shell script, this gives substantial flexibility at the expense of a nice interface. For this reason, *xinit* is not intended for end users.

DISPLAY NAMES

From the user's prospective, every X server has a *display name* of the form:

hostname:displaynumber.screennumber

This information is used by the application to determine how it should connect to the server and which screen it should use by default (on displays with multiple monitors):

hostname

The *hostname* specifies the name of the machine to which the display is physically connected. If the hostname is not given, the most efficient way of communicating to a server on the same machine will be used.

displaynumber

The phrase "display" is usually used to refer to collection of monitors that share a common keyboard and pointer (mouse, tablet, etc.). Most workstations tend to only have one keyboard, and therefore, only one display. Larger, multi-user systems, however, will frequently have several displays so that more than one person can be doing graphics work at once. To avoid confusion, each display on a machine is assigned a *display number* (beginning at 0) when the X server for that display is started. The display number must always be given in a display name.

screennumber

Some displays share a single keyboard and pointer among two or more monitors. Since each monitor has its own set of windows, each screen is assigned a *screen number* (beginning at 0) when the X server for that display is started. If the screen number is not given, then screen 0 will be used.

On POSIX systems, the default display name is stored in your DISPLAY environment variable. This variable is set automatically by the *xterm* terminal emulator. However, when you log into another machine on a network, you'll need to set DISPLAY by hand to point to your display. For example,

```
% setenv DISPLAY myws:0
$ DISPLAY=myws:0; export DISPLAY
```

Finally, most X programs accept a command line option of **-display** *displayname* to temporarily override the contents of DISPLAY. This is most commonly used to pop windows on another person's screen or as part of a "remote shell" command to start an xterm pointing back to your display. For example,

```
% xeyes -display joesws:0 -geometry 1000x1000+0+0
% rsh big xterm -display myws:0 -ls </dev/null &
```

X servers listen for connections on a variety of different communications channels (network byte streams, shared memory, etc.). Since there can be more than one way of contacting a given server, The *hostname* part of the display name is used to determine the type of channel (also called a transport layer) to be used. The sample servers from MIT support the following types of connections:

local

The hostname part of the display name should be the empty string. For example: *:0*, *:1*, and *:0.1*. The most efficient local transport will be chosen.

TCP/IP

The hostname part of the display name should be the server machine's IP address name. Full Internet names, abbreviated names, and IP addresses are all allowed. For example: *expo.lcs.mit.edu:0*, *expo:0*, *18.30.0.212:0*, *bigmachine:1*, and *hydra:0.1*.

DECnet

The hostname part of the display name should be the server machine's nodename followed by two colons instead of one. For example: *myws::0*, *big::1*, and *hydra::0.1*.

ACCESS CONTROL

The sample server provides two types of access control: an authorization protocol which provides a list of ‘‘magic cookies’’ clients can send to request access, and a list of hosts from which connections are always accepted. *Xdm* initializes magic cookies in the server, and also places them in a file accessible to the user. Normally, the list of hosts from which connections are always accepted should be empty, so that only clients with are explicitly authorized can connect to the display. When you add entries to the host list (with *xhost*), the server no longer performs any authorization on connections from those machines. Be careful with this.

The file for authorization which both *xdm* and *Xlib* use can be specified with the environment variable **XAUTHORITY**, and defaults to the file **.Xauthority** in the home directory. *Xdm* uses **$HOME/.Xauthority** and will create it or merge in authorization records if it already exists when a user logs in.

To manage a collection of authorization files containing a collection of authorization records use *xauth*. This program allows you to extract records and insert them into other files. Using this, you can send authorization to remote machines when you login. As the files are machine-independent, you can also simply copy the files or use NFS to share them. If you use several machines, and share a common home directory with NFS, then you never really have to worry about authorization files, the system should work correctly by default. Note that magic cookies transmitted ''in the clear'' over NFS or using *ftp* or *rcp* can be ''stolen'' by a network eavesdropper, and as such may enable unauthorized access. In many environments this level of security is not a concern, but if it is, you need to know the exact semantics of the particular magic cookie to know if this is actually a problem.

GEOMETRY SPECIFICATIONS

One of the advantages of using window systems instead of hardwired terminals is that applications don't have to be restricted to a particular size or location on the screen. Although the layout of windows on a display is controlled by the window manager that the user is running (described below), most X programs accept a command line argument of the form **-geometry** *WIDTHxHEIGHT+XOFF+YOFF* (where *WIDTH*, *HEIGHT*, *XOFF*, and *YOFF* are numbers) for specifying a preferred size and location for this application's main window.

The *WIDTH* and *HEIGHT* parts of the geometry specification are usually measured in either pixels or characters, depending on the application. The *XOFF* and *YOFF* parts are measured in pixels and are used to specify the distance of the window from the left or right and top and bottom edges of the screen, respectively. Both types of offsets are measured from the indicated edge of the screen to the corresponding edge of the window. The X offset may be specified in the following ways:

+XOFF The left edge of the window is to be placed *XOFF* pixels in from the left edge of the screen (i.e. the X coordinate of the window's origin will be *XOFF*). *XOFF* may be negative, in which case the window's left edge will be off the screen.

-XOFF The right edge of the window is to be placed *XOFF* pixels in from the right edge of the screen. *XOFF* may be negative, in which case the window's right edge will be off the screen.

The Y offset has similar meanings:

+YOFF The top edge of the window is to be *YOFF* pixels below the top edge of the screen (i.e. the Y coordinate of the window's origin will be *YOFF*). *YOFF* may be negative, in which case the window's top edge will be off the screen.

-YOFF The bottom edge of the window is to be *YOFF* pixels above the bottom edge of the screen. *YOFF* may be negative, in which case the window's bottom edge will be off the screen.

Offsets must be given as pairs; in other words, in order to specify either *XOFF* or *YOFF* both must be present. Windows can be placed in the four corners of the screen using the following specifications:

+0+0	upper left hand corner.
-0+0	upper right hand corner.
-0-0	lower right hand corner.
+0-0	lower left hand corner.

In the following examples, a terminal emulator will be placed in roughly the center of the screen and a load average monitor, mailbox, and clock will be placed in the upper right hand corner:

```
xterm -fn 6x10 -geometry 80x24+30+200 &
xclock -geometry 48x48-0+0 &
xload -geometry 48x48-96+0 &
xbiff -geometry 48x48-48+0 &
```

WINDOW MANAGERS

The layout of windows on the screen is controlled by special programs called *window managers*. Although many window managers will honor geometry specifications as given, others may choose to ignore them (requiring the user to explicitly draw the window's region on the screen with the pointer, for example).

Since window managers are regular (albeit complex) client programs, a variety of different user interfaces can be built. The core distribution comes with a window manager named *twm* which supports overlapping windows, popup menus, point-and-click or click-to-type input models, title bars, nice icons (and an icon manager for those who don't like separate icon windows).

Several other window managers are available in the user-contributed software: *gwm*, *m_swm*, *olwm*, and *tekwm*.

FONT NAMES

Collections of characters for displaying text and symbols in X are known as *fonts*. A font typically contains images that share a common appearance and look nice together (for example, a single size, boldness, slant, and character set). Similarly, collections of fonts that are based on a common type face (the variations are usually called roman, bold, italic, bold italic, oblique, and bold oblique) are called *families*.

Sets of font families of the same resolution (usually measured in dots per inch) are further grouped into *directories* (so named because they were initially stored in file system directories). Each directory contains a database which lists the name of the font and information on how to find the font. The server uses these databases to translate *font names* (which have nothing to do with file names) into font data.

The list of font directories in which the server looks when trying to find a font is controlled by the *font path*. Although most installations will choose to have the server start up with all of the commonly used font directories, the font path can be changed at any time with the *xset* program. However, it is important to remember that the directory names are on the **server**'s machine, not on the application's.

The default font path for the sample server contains three directories:

/usr/lib/X11/fonts/misc
: This directory contains many miscellaneous fonts that are useful on all systems. It contains a small family of fixed-width fonts in pixel heights 5 through 10, a family of fixed-width fonts from Dale Schumacher in similar pixel heights, several Kana fonts from Sony Corporation, a Kanji font, the standard cursor font, two cursor fonts from Digital Equipment Corporation, and OPEN LOOK(tm) cursor and glyph fonts from Sun Microsystems. It also has font name aliases for the font names **fixed** and **variable**.

/usr/lib/X11/fonts/75dpi
: This directory contains fonts contributed by Adobe Systems, Inc., Digital Equipment Corporation, Bitstream, Inc., Bigelow and Holmes, and Sun Microsystems, Inc. for 75 dots per inch

displays. An integrated selection of sizes, styles, and weights are provided for each family.

/usr/lib/X11/fonts/100dpi

This directory contains 100 dots per inch versions of some of the fonts in the *75dpi* directory.

Font databases are created by running the *mkfontdir* program in the directory containing the source or compiled versions of the fonts (in both compressed and uncompressed formats). Whenever fonts are added to a directory, *mkfontdir* should be rerun so that the server can find the new fonts. To make the server reread the font database, reset the font path with the *xset* program. For example, to add a font to a private directory, the following commands could be used:

```
% cp newfont.snf ~/myfonts
% mkfontdir ~/myfonts
% xset fp rehash
```

The *xlsfonts* program can be used to list all of the fonts that are found in font databases in the current font path. Font names tend to be fairly long as they contain all of the information needed to uniquely identify individual fonts. However, the sample server supports wildcarding of font names, so the full specification

-adobe-courier-medium-r-normal--10-100-75-75-m-60-iso8859-1

could be abbreviated as:

--courier-medium-r-normal--*-100-*-*-*-*-*-**

or, more tersely (but less accurately):

-courier-medium-r-normal--*-100-

Because the shell also has special meanings for * and *?*, wildcarded font names should be quoted:

```
% xlsfonts -fn '*-courier-medium-r-normal--*-100-*'
```

If more than one font in a given directory in the font path matches a wildcarded font name, the choice of which particular font to return is left to the server. However, if fonts from more than one directory match a name, the returned font will always be from the first such directory in the font path. The example given above will match fonts in both the *75dpi* and *100dpi* directories; if the *75dpi* directory is ahead of the *100dpi* directory in the font path, the smaller version of the font will be used.

COLOR NAMES

Most applications provide ways of tailoring (usually through resources or command line arguments) the colors of various elements in the text and graphics they display. Although black and white displays don't provide much of a choice, color displays frequently allow anywhere between 16 and 16 million different colors.

Colors are usually specified by their commonly-used names (for example, *red*, *white*, or *medium slate blue*). The server translates these names into appropriate screen colors using a color database that can usually be found in */usr/lib/X11/rgb.txt*. Color names are case-insensitive, meaning that *red*, *Red*, and *RED* all refer to the same color.

Many applications also accept color specifications of the following form:

```
#rgb
#rrggbb
#rrrgggbbb
#rrrrggggbbbb
```

where *r*, *g*, and *b* are hexadecimal numbers indicating how much *red*, *green*, and *blue* should be displayed (zero being none and ffff being on full). Each field in the specification must have the same number of digits (e.g., #rrgb or #gbb are not allowed). Fields that have fewer than four digits (e.g. #rgb) are padded out with zero's following each digit (e.g. #r000g000b000). The eight primary colors can be represented as:

black	#000000000000 (no color at all)
red	#ffff00000000
green	#0000ffff0000
blue	#00000000ffff
yellow	#ffffffff0000 (full red and green, no blue
magenta	#ffff0000ffff
cyan	#0000ffffffff
white	#ffffffffffff (full red, green, and blue)

Unfortunately, RGB color specifications are highly unportable since different monitors produce different shades when given the same inputs. Similarly, color names aren't portable because there is no standard naming scheme and because the color database needs to be tuned for each monitor.

Application developers should take care to make their colors tailorable.

KEYS

The X keyboard model is broken into two layers: server-specific codes (called *keycodes*) which represent the physical keys, and server-independent symbols (called *keysyms*) which represent the letters or words that appear on the keys. Two tables are kept in the server for converting keycodes to keysyms:

modifier list

Some keys (such as Shift, Control, and Caps Lock) are known as *modifier* and are used to select different symbols that are attached to a single key (such as Shift-a generates a capital A, and Control-l generates a formfeed character ^L). The server keeps a list of keycodes corresponding to the various modifier keys. Whenever a key is pressed or released, the server generates an *event* that contains the keycode of the indicated key as well as a mask that specifies which of the modifier keys are currently pressed. Most servers set up this list to initially contain the various shift, control, and shift lock keys on the keyboard.

keymap table

Applications translate event keycodes and modifier masks into keysyms using a *keysym table* which contains one row for each keycode and one column for various modifier states. This table is initialized by the server to correspond to normal typewriter conventions, but is only used by client programs.

Although most programs deal with keysyms directly (such as those written with the X Toolkit Intrinsics), most programming libraries provide routines for converting keysyms into the appropriate type of string (such as ISO Latin-1).

OPTIONS

Most X programs attempt to use the same names for command line options and arguments. All applications written with the X Toolkit Intrinsics automatically accept the following options:

–display *display*

This option specifies the name of the X server to use.

–geometry *geometry*

This option specifies the initial size and location of the window.

–bg *color*, **–background** *color*

Either option specifies the color to use for the window background.

–bd *color*, **–bordercolor** *color*
Either option specifies the color to use for the window border.

–bw *number*, **–borderwidth** *number*
Either option specifies the width in pixels of the window border.

–fg *color*, **–foreground** *color*
Either option specifies the color to use for text or graphics.

–fn *font*, **-font** *font*
Either option specifies the font to use for displaying text.

–iconic
This option indicates that the user would prefer that the application's windows initially not be visible as if the windows had be immediately iconified by the user. Window managers may choose not to honor the application's request.

–name
This option specifies the name under which resources for the application should be found. This option is useful in shell aliases to distinguish between invocations of an application, without resorting to creating links to alter the executable file name.

–rv, **–reverse**
Either option indicates that the program should simulate reverse video if possible, often by swapping the foreground and background colors. Not all programs honor this or implement it correctly. It is usually only used on monochrome displays.

+rv
This option indicates that the program should not simulate reverse video. This is used to override any defaults since reverse video doesn't always work properly.

–selectionTimeout
This option specifies the timeout in milliseconds within which two communicating applications must respond to one another for a selection request.

–synchronous
This option indicates that requests to the X server should be sent synchronously, instead of asynchronously. Since *Xlib* normally buffers requests to the server, errors do not necessarily get reported immediately after they occur. This option turns off the buffering so that the application can be debugged. It should never be used with a working program.

–title *string*

This option specifies the title to be used for this window. This information is sometimes used by a window manager to provide some sort of header identifying the window.

–xnllanguage *language[_territory][.codeset]*

This option specifies the language, territory, and codeset for use in resolving resource and other filenames.

–xrm *resourcestring*

This option specifies a resource name and value to override any defaults. It is also very useful for setting resources that don't have explicit command line arguments.

RESOURCES

To make the tailoring of applications to personal preferences easier, X supports several mechanisms for storing default values for program resources (e.g. background color, window title, etc.) Resources are specified as strings of the form

*appname*subname*subsubname...: value*

that are read in from various places when an application is run. By convention, the application name is the same as the program name, but with the first letter capitalized (e.g. *Bitmap* or *Emacs*) although some programs that begin with the letter "x" also capitalize the second letter for historical reasons. The precise syntax for resources is:

```
ResourceLine        =Comment | ResourceSpec
Comment             ="!" string | <empty line>
ResourceSpec        =WhiteSpace ResourceName WhiteSpace ":" WhiteSpace value
ResourceName        =[Binding] ComponentName {Binding ComponentName}
Binding             ="." | "*"
WhiteSpace          ={" " | "\t"}
ComponentName       ={"a"-"z" | "A"-"Z" | "0"-"9" | "_" | "-"}
value               =string
string              ={<any character not including "\n">}
```

Note that elements enclosed in curly braces ({...}) indicate zero or more occurrences of the enclosed elements

To allow values to contain arbitrary octets, the 4-character sequence \nnn, where n is a digit in the range of "0"–"7", is recognized and replaced with a single byte that contains this sequence interpreted as an octal number. For example, a value containing a NULL byte can be stored by specifying "\000".

The *Xlib* routine *XGetDefault(3X)* and the resource utilities within Xlib and the X Toolkit Intrinsics obtain resources from the following sources:

RESOURCE_MANAGER root window property
: Any global resources that should be available to clients on all machines should be stored in the RESOURCE_MANAGER property on the root window using the *xrdb* program. This is frequently taken care of when the user starts up X through the display manager or *xinit.*

application-specific files
: Programs that use the X Toolkit Intrinsics will also look in the directories named by the environment variable XUSER-FILESEARCHPATH or the environment variable XAPPLRES-DIR, plus directories in a standard place (usually under /usr/lib/X11/, but this can be overridden with the XFILESEAR-CHPATH environment variable) for application-specific resources. See the *X Toolkit Intrinsics - C Language Interface* manual for details.

XENVIRONMENT
: Any user- and machine-specific resources may be specified by setting the XENVIRONMENT environment variable to the name of a resource file to be loaded by all applications. If this variable is not defined, a file named *$HOME/*.Xdefaults-*hostname* is looked for instead, where *hostname* is the name of the host where the application is executing.

–xrm *resourcestring*
: Applications that use the X Toolkit Intrinsics can have resources specified from the command line. The *resourcestring* is a single resource name and value as shown above. Note that if the string contains characters interpreted by the shell (e.g., asterisk), they must be quoted. Any number of **–xrm** arguments may be given on the command line.

Program resources are organized into groups called *classes*, so that collections of individual resources (each of which are called *instances*) can be set all at once. By convention, the instance name of a resource begins with a lowercase letter and class name with an upper case letter. Multiple word resources are concatenated with the first letter of the succeeding words capitalized. Applications written with the X Toolkit Intrinsics will have at least the following resources:

background (class Background)
This resource specifies the color to use for the window background.

borderWidth (class BorderWidth)
This resource specifies the width in pixels of the window border.

borderColor (class BorderColor)
This resource specifies the color to use for the window border.

Most applications using the X Toolkit Intrinsics also have the resource **foreground** (class **Foreground**), specifying the color to use for text and graphics within the window.

By combining class and instance specifications, application preferences can be set quickly and easily. Users of color displays will frequently want to set Background and Foreground classes to particular defaults. Specific color instances such as text cursors can then be overridden without having to define all of the related resources. For example,

```
bitmap*Dashed: off
XTerm*cursorColor: gold
XTerm*multiScroll: on
XTerm*jumpScroll: on
XTerm*reverseWrap: on
XTerm*curses: on
XTerm*Font: 6x10
XTerm*scrollBar: on
XTerm*scrollbar*thickness: 5
XTerm*multiClickTime: 500
XTerm*charClass: 33:48,37:48,45-47:48,64:48
XTerm*cutNewline: off
XTerm*cutToBeginningOfLine: off
XTerm*titeInhibit: on
XTerm*ttyModes: intr ^c erase ^? kill ^u
XLoad*Background: gold
XLoad*Foreground: red
XLoad*highlight: black
XLoad*borderWidth: 0
emacs*Geometry: 80x65-0-0
emacs*Background: #5b7686
emacs*Foreground: white
emacs*Cursor: white
emacs*BorderColor: white
emacs*Font: 6x10
xmag*geometry: -0-0
```

```
        xmag*borderColor:  white
```

If these resources were stored in a file called *.Xresources* in your home directory, they could be added to any existing resources in the server with the following command:

```
        % xrdb -merge $HOME/.Xresources
```

This is frequently how user-friendly startup scripts merge user-specific defaults into any site-wide defaults. All sites are encouraged to set up convenient ways of automatically loading resources. See the *Xlib* manual section *Using the Resource Manager* for more information.

EXAMPLES

The following is a collection of sample command lines for some of the more frequently used commands. For more information on a particular command, please refer to that command's manual page.

```
        % xrdb -load $HOME/.Xresources
        % xmodmap -e "keysym BackSpace = Delete"
        % mkfontdir /usr/local/lib/X11/otherfonts
        % xset fp+ /usr/local/lib/X11/otherfonts
        % xmodmap $HOME/.keymap.km
        % xsetroot -solid '#888'
        % xset b 100 400 c 50 s 1800 r on
        % xset q
        % twm
        % xmag
        % xclock -geometry 48x48-0+0 -bg blue -fg white
        % xeyes -geometry 48x48-48+0
        % xbiff -update 20
        % xlsfonts '*helvetica*'
        % xlswins -l
        % xwininfo -root
        % xdpyinfo -display joesworkstation:0
        % xhost -joesworkstation
        % xrefresh
        % xwd | xwud
        % bitmap companylogo.bm 32x32
        % xcalc -bg blue -fg magenta
        % xterm -geometry 80x66-0-0 -name myxterm $*
```

DIAGNOSTICS

A wide variety of error messages are generated from various programs. Various toolkits are encouraged to provide a common mechanism for locating error text so that applications can be tailored easily. Programs written to interface directly to the *Xlib* C language library are expected to do their own error checking.

The default error handler in *Xlib* (also used by many toolkits) uses standard resources to construct diagnostic messages when errors occur. The defaults for these messages are usually stored in */usr/lib/X11/XErrorDB*. If this file is not present, error messages will be rather terse and cryptic.

When the X Toolkit Intrinsics encounter errors converting resource strings to the appropriate internal format, no error messages are usually printed. This is convenient when it is desirable to have one set of resources across a variety of displays (e.g. color vs. monochrome, lots of fonts vs. very few, etc.), although it can pose problems for trying to determine why an application might be failing. This behavior can be overridden by the setting the *StringConversionsWarning* resource.

To force the X Toolkit Intrinsics to always print string conversion error messages, the following resource should be placed at the top of the file that gets loaded onto the RESOURCE_MANAGER property using the *xrdb* program (frequently called *.Xresources* or *.Xres* in the user's home directory):

```
*StringConversionWarnings: on
```

To have conversion messages printed for just a particular application, the appropriate instance name can be placed before the asterisk:

```
xterm*StringConversionWarnings: on
```

BUGS

If you encounter a **repeatable** bug, please contact your site administrator for instructions on how to submit an X Bug Report.

SEE ALSO

XConsortium(1), XStandards(1), appres(1), bdftosnf(1), bitmap(1), imake(1), listres(1), maze(1), mkfontdir(1), muncher(1), oclock(1), puzzle(1), resize(1), showsnf(1), twm(1), xauth(1), xbiff(1), xcalc(1), xclipboard(1), xclock(1), xditview(1), xdm(1), xdpyinfo(1), xedit(1), xev(1), xeyes(1), xfd(1), xfontsel(1), xhost(1), xinit(1), xkill(1), xload(1), xlogo(1), xlsatoms(1), xlsclients(1), xlsfonts(1), xlswins(1), xmag(1), xman(1), xmh(1), xmodmap(1), xpr(1), xprop(1), xrdb(1), xrefresh(1), xset(1), xsetroot(1), xstdcmap(1), xterm(1), xwd(1), xwininfo(1), xwud(1), Xserver(1), Xapollo(1), Xcfbpmax(1), Xhp(1), Xibm(1), XmacII(1), Xmfbpmax(1), Xqdss(1), Xqvss(1), Xsun(1), Xtek(1), kbd_mode(1), todm(1), tox(1), biff(1), init(8), ttys(5), *Xlib – C Language X Interface*, *X Toolkit Intrinsics - C Language Interface*, and *Using and Specifying X Resources*

COPYRIGHT

TRADEMARKS

UNIX and OPEN LOOK are trademarks of AT&T. X Window System is a trademark of MIT.

AUTHORS

A cast of thousands, literally. The X distribution is brought to you by the MIT X Consortium. The staff members at MIT responsible for this release are: Donna Converse (MIT X Consortium), Jim Fulton (MIT X Consortium), Michelle Leger (MIT X Consortium), Keith Packard (MIT X Consortium), Chris Peterson (MIT X Consortium), Bob Scheifler (MIT X Consortium), and Ralph Swick (Digital/MIT Project Athena).

NAME

x10tox11 – X version 10 to version 11 protocol converter

SYNOPSIS

x10tox11 [-display host:display]

DESCRIPTION

x10tox11 masquerades as an X Window System Version 10 server. It enables an X Version 10 client to run unchanged under X Version 11 by converting Version 10 requests into appropriate Version 11 requests, and by converting all Version 11 events received from the server into Version 10 events. From the perspective of Version 10 clients, all Version 11 clients look like Version 10 clients; and from the perspective of Version 11 clients, all Version 10 clients just look like Version 11 clients. Hence, a Version 11 window manager can manipulate Version 10 clients.

This program does NOT use the X10 *libnest* ddX library. It does actual protocol translation, rather than simply using X11 graphics calls to implement X10 low level operations. As a result, it is both faster and more robust than the X10 Xnest server.

TYPICAL USAGE

The protocol converter must be run after the X11 server is running and should be run in the background:

x10tox11 &

The program will continue to run until you intentionally kill it or the X11 server is shut down.

OPTIONS

-display host:display

Standard option for specifying the X11 display to which you wish to be connected. By default, it uses unix:0.0. Note that *x10tox11* will always pretend to be an X10 server with the same display number as the X11 server to which it connects. For example, if the DISPLAY environment variable or the *-display* option specifies *fizzle:1.0*, then *x10tox11* will connect to the X11 server on host *fizzle* for display 1 and then will pretend to the the X10 server for display 1. Consequently, your X10 clients will expect to have the environment variable DISPLAY set to *fizzle:1* (but they should still work even if your X10 clients use *fizzle:1.0*).

MinimumTileSize=n

Set minimum acceptable tile size to *n*. There is a difference in semantics between X10's XQueryShape and X11's XQueryBest-Size such that X11 will allow any tile size but will return the optimum whereas X10 enforced a minimum tile size. Usually this minimum tile size was 16 and this is the default for *x10tox11*. If you find that this makes your X10 clients break, then you can override it with this option.

help

This prints out a usage message and exits.

NoOverrideRedirect

This instructs *x10tox11* to make every effort not to use OverrideRedirect when creating and mapping windows. Normally, *x10tox11* creates all windows with the OverrideRedirect attribute set to true. Placing this option on the command line will cause *x10tox11* not to use OverrideRedirect except for windows that look like they might be menus. This will allow window managers that provide title bars to do so. Unfortunately, it is impossible to determine ahead of time what an X10 client intends to do with windows. In addition, X10 clients are known to spontaneously unmap their windows which upsets X11 window managers unless the OverrideRedirect attribute is true. Further, some X11 window managers may refuse to resize or move windows that are marked with OverrideRedirect. This may can be fixed to some extent when an Inter Client Communications Convention Manual (ICCCM) is adopted by the X11 community.

SEE ALSO

X(1), Xserver(1)

BUGS

There are limitations with respect to emulating Version 10 through a Version 11 server. See the file /usr/lib/X/x10tox11.help for more details.

Some window managers may refuse to move, resize or perform any operations on X10 client windows because, by default,

If the source is compiled with certain flags, there are significant debugging facilities available. Using the *help* option will tell you whether debugging facilities are available. *x10tox11* marks them with OverrideRedirect. See **OPTIONS** above.

COPYRIGHT

AUTHOR

Todd Brunhoff, Visual Systems Laboratory, Tektronix.

NAME

xargs – construct argument list(s) and execute command

SYNOPSIS

xargs [flags] [command [initial-arguments]]

DESCRIPTION

xargs combines the fixed *initial-arguments* with arguments read from standard input to execute the specified *command* one or more times. The number of arguments read for each *command* invocation and the manner in which they are combined are determined by the flags specified.

command, which may be a shell file, is searched for, using one's **$PATH**. If *command* is omitted, **/bin/echo** is used.

Arguments read in from standard input are defined to be contiguous strings of characters delimited by one or more blanks, tabs, or new-lines; empty lines are always discarded. Blanks and tabs may be embedded as part of an argument if escaped or quoted. Characters enclosed in quotes (single or double) are taken literally, and the delimiting quotes are removed. Outside of quoted strings a backslash (\) will escape the next character.

Each argument list is constructed starting with the *initial-arguments*, followed by some number of arguments read from standard input (Exception: see **–i** flag). Flags **–i**, **–l**, and **–n** determine how arguments are selected for each command invocation. When none of these flags are coded, the *initial-arguments* are followed by arguments read continuously from standard input until an internal buffer is full, and then *command* is executed with the accumulated args. This process is repeated until there are no more args. When there are flag conflicts (e.g., **–l** vs. **–n**), the last flag has precedence. *Flag* values are:

–l*number* *command* is executed for each non-empty *number* lines of arguments from standard input. The last invocation of *command* will be with fewer lines of arguments if fewer than *number* remain. A line is considered to end with the first new-line *unless* the last character of the line is a blank or a tab; a trailing blank/tab signals continuation through the next non-empty line. If *number* is omitted, 1 is assumed. Option **–x** is forced.

–i*replstr* Insert mode: *command* is executed for each line from standard input, taking the entire line as a single arg, inserting it in *initial-arguments* for each occurrence of *replstr*. A maximum of 5 arguments in *initial-arguments* may each contain one or more instances of *replstr*. Blanks and tabs at the beginning of each line are thrown away. Constructed arguments may not grow larger than 255 characters, and option **−x** is also forced. {} is assumed for *replstr* if not specified.

−n*number* Execute *command* using as many standard input arguments as possible, up to *number* arguments maximum. Fewer arguments will be used if their total size is greater than *size* characters, and for the last invocation if there are fewer than *number* arguments remaining. If option **−x** is also coded, each *number* arguments must fit in the *size* limitation, else *xargs* terminates execution.

−t Trace mode: The *command* and each constructed argument list are echoed to file descriptor 2 just prior to their execution.

−p Prompt mode: The user is asked whether to execute *command* each invocation. Trace mode (**−t**) is turned on to print the command instance to be executed, followed by a ?... prompt. A reply of y (optionally followed by anything) will execute the command; anything else, including just a carriage return, skips that particular invocation of *command*.

−x Causes *xargs* to terminate if any argument list would be greater than *size* characters; **−x** is forced by the options **−i** and **−l**. When neither of the options **−i**, **−l**, or **−n** are coded, the total length of all arguments must be within the *size* limit.

−s*size* The maximum total size of each argument list is set to *size* characters; *size* must be a positive integer less than or equal to 470. If **−s** is not coded, 470 is taken as the default. Note that the character count for *size* includes one extra character for each argument and the count of characters in the command name.

−e*eofstr* *eofstr* is taken as the logical end-of-file string. Underbar (_) is assumed for the logical EOF string if **−e** is not coded. The value **−e** with no *eofstr* coded turns off the logical EOF string capability (underbar is taken literally). *xargs* reads standard input until either end-of-file or the logical

EOF string is encountered.

xargs will terminate if either it receives a return code of **−1** from, or if it cannot execute, *command*. When *command* is a shell program, it should explicitly *exit* (see *sh*(1)) with an appropriate value to avoid accidentally returning with **−1**.

EXAMPLES

The following will move all files from directory $1 to directory $2, and echo each move command just before doing it:

```
ls $1 | xargs –i –t mv $1/{ } $2/{ }
```

The following will combine the output of the parenthesized commands onto one line, which is then echoed to the end of file *log*:

```
(logname; date; echo $0 $*) | xargs >>log
```

The user is asked which files in the current directory are to be archived and archives them into *arch* (1.) one at a time, or (2.) many at a time.

```
1. ls | xargs –p –l ar r arch
2. ls | xargs –p –l | xargs ar r arch
```

The following will execute *diff*(1) with successive pairs of arguments originally typed as shell arguments:

```
echo $* | xargs –n2 diff
```

SEE ALSO

sh(1).

NAME

xauth - X authority file utility

SYNOPSIS

xauth [-f *authfile*] [-vqib] [*command arg...*]

DESCRIPTION

The *xauth* program is used to edit and display the authorization information used in connecting to the X server. This program is usually to extract authorization records from one machine and merge them in on another (as is the case when using remote logins or to grant access to other users). Commands (described below) may be entered interactively, on the *xauth* command line, or in scripts. Note that this program does **not** contact the X server.

OPTIONS

The following options may be used with *xauth*. They may be given indivually (e.g. *–q –i*) or may combined (e.g. *-qi*):

–f *authfile*
: This option specifies the name of the authority file to use. By default, *xauth* will use the file specified by the XAUTHORITY environment variable or *Xauthority* in the user's home directory.

–q
: This option indicates that *xauth* should operate quietly and not print unsolicited status messages. This is the default if an *xauth* command is is given on the command line or if the standard output is not directed to a terminal.

–v
: This option indicates that *xauth* should operate verbosely and print status messages indicating the results of various operations (e.g. how many records have been read in or written out). This is the default if *xauth* is reading commands from its standard input and its standard output is directed to a terminal.

–i
: This option indicates that *xauth* should ignore any authority file locks. Normally, *xauth* will refuse to read or edit any authority files that have been locked by other programs (usually *xdm* or another *xauth*).

–b
: This option indicates that *xauth* should attempt to break any authority file locks before proceeding and should only be used to clean up stale locks.

COMMANDS

The following commands may be used to manipulate authority files:

add *displayname protocolname hexkey*
An authorization entry for the indicated display using the given protocol and key data is added to the authorization file. The data is specified as an even-lengthed string of hexidecimal digits, each pair representing one octet. The first digit gives the most significant 4 bits of the octet and the second digit gives the least significant 4 bits. A protocol name consisting of just a single period is treated as an abbreviation for *MIT-MAGIC-COOKIE-1*.

[n]extract *filename displayname...*
Authorization entries for each of the specified displays are written to the indicated file. If the *nextract* command is used, the entries are written in a numeric format suitable for non-binary transmission (such as secure electronic mail). The extracted entries can be read back in using the *merge* and *nmerge* commands. If the the filename consists of just a single dash, the entries will be written to the standard output.

[n]list [*displayname...*]
Authorization entries for each of the specified displays (or all if no displays are named) are printed on the standard output. If the *nlist* command is used, entries will be shown in the numeric format used by the *nextract* command; otherwise, they are shown in a textual format. Key data is always displayed in the hexidecimal format given in the description of the *add* command.

[n]merge [*filename...*]
Authorization entries are read from the specified files and are merged into the authorization database, superceeding any matching existing entries. If the *nmerge* command is used, the numeric format given in the description of the *extract* command is used. If a filename consists of just a single dash, the standard input will be read if it hasn't been read before.

remove *displayname...*
Authorization entries matching the specified displays are removed from the authority file.

source *filename*
The specified file is treated as a script containing *xauth* commands to execute. Blank lines and lines beginning with a sharp sign (#) are ignored. A single dash may be used to indicate the standard input, if it hasn't already been read.

info Information describing the authorization file, whether or not any changes have been made, and from where *xauth* commands are being read is printed on the standard output.

exit If any modifications have been made, the authority file is written out (if allowed), and the program exits. An end of file is treated as an implicit *exit* command.

quit The program exits, ignoring any modifications. This may also be accomplished by pressing the interrupt character.

help [*string*]
A description of all commands that begin with the given string (or all commands if no string is given) is printed on the standard output.

? A short list of the valid commands is printed on the standard output.

DISPLAY NAMES

Display names for the *add*, *[n]extract*, *[n]list*, *[n]merge*, and *remove* commands use the same format as the DISPLAY environment variable and the common *-display* command line argument. Display-specific information (such as the screen number) is unnecessary and will be ignored. Same-machine connections (such as UNIX-domain sockets, shared memory, and the Internet Protocol hostname *localhost*) are refered to as *hostname*/unix:*displaynumber* so that local entries for different machines may be stored in one authority file.

EXAMPLE

The most common use for *xauth* is to extract the entry for the current display, copy it to another machine, and merge it into the user's authority file on the remote machine:

```
% xauth extract - $DISPLAY | rsh other xauth merge -
```

ENVIRONMENT

This *xauth* program uses the following environment variables:

XAUTHORITY
to get the name of the authority file to use if the *–f* option isn't used. If this variable is not set, *xauth* will use *Xauthority* in the user's home directory.

HOME to get the user's home directory if XAUTHORITY isn't defined.

BUGS

Users that have unsecure networks should take care to use encrypted file transfer mechanisms to copy authorization entries between machines. Similarly, the *MIT-MAGIC-COOKIE-1* protocol is not very useful in unsecure environments. Sites that are interested in additional security may need to use encrypted authorization mechanisms such as Kerberos.

Spaces are currently not allowed in the protocol name. Quoting could be added for the truly perverse.

COPYRIGHT

Copyright 1989, Massachusetts Institute of Technology.
See *X(1)* for a full statement of rights and permissions.

AUTHOR

Jim Fulton, MIT X Consortium

NAME

xbiff - mailbox flag for X

SYNOPSIS

xbiff [-*toolkitoption* ...] [-option ...]

DESCRIPTION

The *xbiff* program displays a little image of a mailbox. When there is no mail, the flag on the mailbox is down. When mail arrives, the flag goes up and the mailbox beeps. By default, pressing any mouse button in the image forces *xbiff* to remember the current size of the mail file as being the ''empty'' size and to lower the flag.

This program is nothing more than a wrapper around the Athena *Mailbox* widget.

OPTIONS

Xbiff accepts all of the standard X Toolkit command line options along with the additional options listed below:

–help This option indicates that a brief summary of the allowed options should be printed on the standard error.

–update *seconds*
This option specifies the frequency in seconds at which *xbiff* should update its display. If the mailbox is obscured and then exposed, it will be updated immediately. The default is 60 seconds.

–file *filename*
This option specifies the name of the file which should be monitored. By default, it watches /usr/spool/mail/*username*, where *username* is your login name.

–volume *percentage*
This option specifies how loud the bell should be rung when new mail comes in.

–shape This option indicates that the mailbox window should be shaped if masks for the empty or full images are given.

The following standard X Toolkit command line arguments are commonly used with *xbiff:*

–display *display*
This option specifies the X server to contact.

–geometry *geometry*
This option specifies the prefered size and position of the mailbox window. The mailbox is 48 pixels wide and 48 pixels high and will be centered in the window.

–bg *color*
This option specifies the color to use for the background of the window. The default is ''white.''

–bd *color*
This option specifies the color to use for the border of the window. The default is ''black.''

–bw *number*
This option specifies the width in pixels of the border surrounding the window.

–fg *color*
This option specifies the color to use for the foreground of the window. The default is ''black.''

–rv This option indicates that reverse video should be simulated by swapping the foreground and background colors.

–xrm *resourcestring*
This option specifies a resource string to be used. This is especially useful for setting resources that do not have separate command line options.

X DEFAULTS

This program uses the *Mailbox* widget in the X Toolkit. It understands all of the core resource names and classes as well as:

checkCommand (class **CheckCommand**)
Specifies a shell command to be executed to check for new mail rather than examining the size of **file**. The specified string value is used as the argument to a *system*(3) call and may therefore contain i/o redirection. An exit status of 0 indicates that new mail is waiting, 1 indicates that there has been no change in size, and 2 indicates that the mail has been cleared.

file (class **File**)
Specifies the name of the file to monitor. The default is to watch /usr/spool/mail/*username*, where *username* is your login name.

onceOnly (class **Boolean**)
Specifies that the bell is only rung the first time new mail is found and is not rung again until at least one interval has passed with no mail waiting. The window will continue to indicate the presence of new mail until it has been retrieved.

width (class **Width**)
Specifies the width of the mailbox.

height (class **Height**)
: Specifies the height of the mailbox.

update (class **Interval**)
: Specifies the frequency in seconds at which the mail should be checked.

volume (class **Volume**)
: Specifies how load the bell should be rung. The default is 33 percent.

foreground (class **Foreground**)
: Specifies the color for the foreground. The default is ``black'' since the core default for background is ``white.''

reverseVideo (class **ReverseVideo**)
: Specifies that the foreground and background should be reversed.

flip (class **Flip**)
: Specifies whether or not the image that is shown when mail has arrived should be inverted. The default is ``true.''

fullPixmap (class **Pixmap**)
: Specifies a bitmap to be shown when new mail has arrived.

emptyPixmap (class **Pixmap**)
: Specifies a bitmap to be shown when no new mail is present.

shapeWindow (class **ShapeWindow**)
: Specifies whether or not the mailbox window should be shaped to the given fullPixmapMask and emptyPixmapMask.

fullPixmapMask (class **PixmapMask**)
: Specifies a mask for the bitmap to be shown when new mail has arrived.

emptyPixmapMask (class **PixmapMask**)
: Specifies a mask for the bitmap to be shown when no new mail is present.

ACTIONS

The *Mailbox* widget provides the following actions for use in event translations:

check() This action causes the widget to check for new mail and display the flag appropriately.

unset() This action causes the widget to lower the flag until new mail comes in.

set() This action causes the widget to raise the flag until the user resets it.

The default translation is

```
<ButtonPress>: unset()
```

ENVIRONMENT

DISPLAY
to get the default host and display number.

XENVIRONMENT
to get the name of a resource file that overrides the global resources stored in the RESOURCE_MANAGER property.

SEE ALSO

X(1), xrdb(1), stat(2)

BUGS

The mailbox bitmaps are ugly.

COPYRIGHT

Copyright 1988, Massachusetts Institute of Technology.
See *X(1)* for a full statement of rights and permissions.

AUTHOR

Jim Fulton, MIT X Consortium
Additional hacks by Ralph Swick, DEC/MIT Project Athena

NAME

xcalc – scientific calculator for X

SYNOPSIS

xcalc [-stipple] [-rpn] [-*toolkitoption...*]

DESCRIPTION

Xcalc is a scientific calculator desktop accessory that can emulate a TI-30 or an HP-10C.

OPTIONS

xcalc accepts all of the standard toolkit command line options along with the additional options listed below:

–stipple This option indicates that the background of the calculator should be drawn using a stipple of the foreground and background colors. On monochrome displays this makes for a nicer display.

–rpn This option indicates that Reverse Polish Notation should be used. In this mode the calculator will look and behave like an HP-10C. Without this flag, it will emulate a TI-30.

OPERATION

Pointer Usage: Most operations are done with the Button1 (usually leftmost button on the pointer). The only exception is that pressing the AC key on the TI calculator or the ON key on the HP calculator with Button3 (usually on the right) will exit the calculator.

Key Usage (Normal mode): The number keys, the +/- key, and the +, -, *, /, and = keys all do exactly what you would expect them to. It should be noted that the operators obey the standard rules of precedence. Thus, entering "3+4*5=" results in "23", not "35". The parentheses can be used to override this. For example, "(1+2+3)*(4+5+6)=" results in "6*15=90". The non-obvious keys are detailed below.

1/x replaces the number in the display with its reciprocal.

x^2 squares the number in the display.

SQRT takes the square root of the number in the display.

CE/C when pressed once, clears the number in the display without clearing the state of the machine. Allows you to re-enter a number if you screw it up. Pressing it twice clears the state, also.

AC clears everything, the display, the state, the memory, everything. Pressing it with the right button 'turns off' the calculator, in that it exits the program. Somewhat more equivalent to throwing the calculator in the trash, if we were to pursue the analogy.

INV inverts the meaning of the function keys. See the individual function keys for details.

sin computes the sine of the number in the display, as interpreted by the current DRG mode (see DRG, below). If inverted, it computes the arcsine.

cos computes the cosine, or arccosine when inverted.

tan computes the tangent, or arctangent when inverted.

DRG changes the DRG mode, as indicated by 'DEG', 'RAD', or 'GRAD' at the bottom of number window of the calculator. When in 'DEG' mode, numbers in the display are taken as being degrees. In 'RAD' mode, numbers are in radians, and in 'GRAD' mode, numbers are in gradians. When inverted, the DRG key has the nifty feature of converting degrees to radians to gradians and vice-versa. Example: put the calculator into 'DEG' mode, and type "45 INV DRG". The display should now show something along the lines of ".785398", which is 45 degrees converted to radians.

e the constant 'e'. (2.7182818...)

EE used for entering exponential numbers. For example, to enter "-2.3E-4" you'd type "2 . 3 +/- EE 4 +/-"

log calculates the log (base 10) of the number in the display. When inverted, it raises "10.0" to the number in the display. For example, typing "3 INV log" should result in "1000".

ln calcuates the log (base e) of the number in the display. When inverted, it raises "e" to the number in the display. For example, typing "e ln" should result in "1"

y^x raises the number on the left to the power of the number on the right. For example "2 y^x 3 =" results in "8", which is 2^3. For a further example, "(1+2+3) y^x (1+2) =" equals "6 y^x 3" which equals "216".

PI the constant 'pi'. (3.1415927....)

x! computes the factorial of the number in the display. The number in the display must be an integer in the range 0-500, though, depending on your math library, it might overflow long before that.

STO copies the number in the display to the memory location.

RCL copies the number from the memory location to the display.

SUM adds the number in the display to the number in the memory location.

EXC swaps the number in the display with the number in the memory location.

Key Usage (RPN mode): The number keys, CHS (change sign), +, -, *, /, and ENTR keys all do exactly what you would expect them to do. Many of the remaining keys are the same as in normal mode. The differences are detailed below.

<- is a backspace key that can be used while typing a number. It will erase digits from the display. Inverse backspace will clear the X register.

ON clears everything, the display, the state, the memory, everything. Pressing it with the right button 'turns off' the calculator, in that it exits the program. Somewhat more equivalent to throwing the calculator in the trash, if we were to pursue the analogy.

INV inverts the meaning of the function keys. This would be the "f" key on an HP calculator, but xcalc does not have the resolution to display multiple legends on each key. See the individual function keys for details.

10^x raises "10.0" to the number in the top of the stack. When inverted, it calculates the log (base 10) of the number in the display.

e^x raises "e" to the number in the top of the stack. When inverted, it calcuates the log (base e) of the number in the display.

STO copies the number in the top of the stack to a memory location. There are 10 memory locations. The desired memory is specified by following this key with pressing a digit key.

RCL pushes the number from the specified memory location onto the stack.

SUM adds the number on top of the stack to the number in the specified memory location.

x:y exchanges the numbers in the top two stack positions, the X and Y registers.

R v rolls the stack downward. When inverted, it rolls the stack upward.

blank these keys were used for programming functions on the HP11-C. Their functionality has not been duplicated here.

KEYBOARD EQUIVALENTS

If you have the pointer in the xcalc window, you can use the keyboard to speed entry, as almost all of the calculator keys have a keyboard equivalent. The number keys, the operator keys, and the parentheses all have the obvious equivalent. The less-obvious equivalents are as follows:

n: +/-	!: x!
p: PI	e: EE
l: ln	^: y^x
i: INV	s: sin
c: cos	t: tan
d: DRG	BS, DEL: CE/C ("<-" in RPN mode)
CR: ENTR	q: quit

COLOR USAGE

Xcalc uses a lot of colors, given the opportunity. In the default case, it will just use two colors (Foreground and Background) for everything. This works out nicely. However, if you're a color fanatic you can specify the colors used for the number keys, the operator (+-*/=) keys, the function keys, the display, and the icon.

X DEFAULTS

stipple Indicates that the background should be stippled. The default is ''on'' for monochrome displays, and ''off'' for color displays.

rpn Specifies that the rpn mode should be used. The default is TI mode.

SEE ALSO

X(1), xrdb(1)

BUGS

HP mode may or may not work correctly.

COPYRIGHT

AUTHORS

John Bradley, University of Pennsylvania
Mark Rosenstein, MIT Project Athena
Donna Converse, MIT X Consortium

NAME

xcalendar - calendar with a notebook for X11

SYNTAX

xcalendar [month [year]]

DESCRIPTION

The *xcalendar* is a simple interactive calendar program with a notebook capability. It is build on the X Toolkit and the Athena Widgets.

If month and year are not provided on the command line they are assumed to be current. To achieve pleasant visual effect you have to load a resource data base to the server. The default database is provided with this program and is also included in this text. Consult the man page for xrdb(1).

INTERACTIONS

Clicking the left mouse button on a day will start a text editor. You can edit and save a text.This text will be associated with the day. You can later on read and edit this text when you open the editor for the same day. The text is saved in a file in the directory ~/Calendar. The editor lets you also clear an entry associated with a particular day.

You can highlight all entries in a month by invoking the function ShowEntries. By default this function is called when the left mouse button is pressed in the title window (where day, month and a year are displayed). Pressing again the same button will unhighlight the entries.

MISSING FEATURES

Currently, to view another month you have to start another process and specify the month and year on the command line. You can run several xcalendars at the same time. However, it would be nice to be able to scroll or browse through the calendar.

To remove all entries in a particular month you have to use your system's commands. The naming scheme for the files makes it easy : the command "rm ~/Calendar/xc*sep1988 " on UNIX(TM) will remove all entries from september 1988. The facility to do that from the xcalendar should be provided.

One can imagine many other useful features. For example automatic parsing of the current day entry in search for appointments to trigger alarms (reminders) at the approriate time. Well, maybe one day...

RESOURCES

The resource data base lets you alter the visual appearance of the program. You can change fonts, border widths, labels, and other resources used by widgets. One use of this facility is to change names of week days and months.

Here are the names of widgets you can use to set various resources:

XCalendar	- class of the application
xcalendar	- top level form
controls	- control panel
quitButton	- quit button
helpButton	- help button
date	- date label
calendar	- calendar frame
daynumbers	- day numbers frame
1-49	- day number buttons
daynames	- day names frame
MON,TUE,WED,THU,FRI,SAT,SUN	- day name buttons
helpWindow	- help window
dayEditor	- editor popup
editorFrame	- editor frame
editorTitle	- editor title
editor	- editor
editorControls	- control panel
doneButton	- done button
saveButton	- save button
clearEntry	- clear entry button

Application specific resources:

reverseVideoMark - if True the entries are highlighted in reverse video; default True for black and white, and False for color displays;

setMarkBackground - if True and reverseVideoMark is False the entries are highlighted by setting background to markBackground ;

markBackground - background color for highlighting entries;

setMarkForeground - analogous to setMarkBackground;

markForeground - foreground color for highlighting entries;

setMarkBackground and *setMarkForeground* can take any combination of values.

january,february,..,december - these resources can be used for changing names of months;

firstDay - an integer between 1-7, indicating the day to start a week with,

```
                    default:  1 (Monday);
markOnStartup  -  if True mark the entries upon startup,
                    default: False;
helpFile      -  full pathname of the xcalendar.hlp file,
                    default: /usr/lib/X11/xcalendar.hlp;
textBufferSize  -  maximum size of the text buffer in the day editor,
                    default: 2048;
```

DEFAULT RESOURCE DATA BASE:

```
XCalendar*Font: vtsingle
XCalendar*Background: wheat
XCalendar*BorderWidth:          2
XCalendar*calendar*borderWidth: 0
XCalendar*controls*borderWidth: 0
XCalendar*date*font:            8x13bold
XCalendar*date*borderWidth: 0
XCalendar*daynames*font:                 8x13bold

XCalendar*daynames*Background: PaleGreen

XCalendar*daynames.SUN*Foreground: Red
XCalendar*daynames.SAT*Foreground: DarkGreen

XCalendar*daynumbers.7*Foreground: Red
XCalendar*daynumbers.14*Foreground: Red
XCalendar*daynumbers.21*Foreground: Red
XCalendar*daynumbers.28*Foreground: Red
XCalendar*daynumbers.35*Foreground: Red
XCalendar*daynumbers.42*Foreground: Red


XCalendar*controls*helpButton*Background: DarkGreen
XCalendar*controls*helpButton*Foreground: wheat
XCalendar*controls*quitButton*Background: DarkGreen
XCalendar*controls*quitButton*Foreground: wheat


XCalendar*daynumbers*Foreground: DarkGreen
XCalendar*editorTitle*Background: PaleGreen
XCalendar*editorTitle*Foreground: Red
XCalendar*editorControls*Background: PaleGreen
XCalendar*editorControls*doneButton*Background: Red
XCalendar*editorControls*doneButton*Foreground: wheat
XCalendar*editorControls*saveButton*Background: Red
```

```
XCalendar*editorControls*saveButton*Foreground: wheat
XCalendar*editorControls*clearEntry*Background: DarkGreen
XCalendar*editorControls*clearEntry*Foreground: wheat

XCalendar*dayEditor*geometry: 300x150
XCalendar*dayEditor*editorTitle*font: 8x13bold
XCalendar*dayEditor*editorTitle*Foreground: DarkGreen

XCalendar*helpWindow*geometry: 600x350
XCalendar*helpWindow*editorTitle*font: 8x13bold

XCalendar*doneButton*Label: done
XCalendar*editorTitle*Label: Help
XCalendar*helpButton*Label: help
XCalendar*quitButton*Label: quit
XCalendar*saveButton*Label: save

XCalendar*markBackground: PaleGreen
XCalendar*setMarkBackground: True
```

FILES

$HOME/Calendar/*

SEE ALSO

xrdb(1)

BUGS

Save button handler in the editor cannot detect when a text is pasted. Workaround : type something to the ditor to activate save button.

AUTHOR

Copyright 1988 by Massachusetts Institute of Technology Roman J. Budzianowski, MIT Project Athena

NAME

xclipboard - X clipboard client

SYNOPSIS

xclipboard [*-toolkitoption* ...] [-w] [-nw]

DESCRIPTION

The *xclipboard* program is used to collect and display text selections that are sent to the CLIP_BOARD by other clients. It is typically used to gather together and hold a block of text that has been selected from a variety of different places.

Since *xclipboard* uses a Text Widget to display the contents of the clipboard, text sent to the CLIP_BOARD may be re-selected for use in other applications.

An *xclipboard* window has the following buttons across the top:

quit When this button is pressed, *xclipboard* exits.

erase When this button is pressed, the contents of the text window are erased.

OPTIONS

The *xclipboard* program accepts all of the standard X Toolkit command line options as well as the following:

–w This option indicates that lines of text that are too long to be displayed on one line in the clipboard should wrap around to the following lines.

–nw This option indicates that long lines of text should not wrap around.

SENDING TO CLIPBOARD

Text is copied to the clipboard whenever a client asserts ownership of the **CLIP_BOARD** selection. Examples of event bindings that a user may wish to include in his/her resource configuration file to use the clipboard are:

```
*VT100.Translations: #override \
        Button1 <Btn2Down>:    select-end(CLIPBOARD) \n\
        Button1 <Btn2Up>:      ignore()

*Text.Translations: #override \
        Button1 <Btn2Down>:    extend-end(CLIPBOARD)
```

X DEFAULTS

This program accepts all of the standard X Toolkit resource names and classes as well as:

wordWrap (class **WordWrap**)
: This resource specifies whether or not lines of text should wrap around to the following lines. The default is *no*.

SEE ALSO

X(1), xcutsel(1), xterm(1), individual client documentation for how to make a selection and send it to the CLIP_BOARD.

BUGS

The erase button is not yet implemented.

It would be nice to have a way of specifying the file in which the clipboard contents are saved.

FILES

/usr/lib/X11/app-defaults/XClipboard - specifies required resources

COPYRIGHT

Copyright 1988, Massachusetts Institute of Technology
See *X(1)* for a full statement of rights and permissions.

AUTHOR

Ralph R. Swick, DEC/MIT Project Athena

NAME

xclock - analog / digital clock for X

SYNOPSIS

xclock [-*toolkitoption* ...] [-option ...]

DESCRIPTION

The *xclock* program displays the time in analog or digital form. The time is continuously updated at a frequency which may be specified by the user. This program is nothing more than a wrapper around the Athena Clock widget.

OPTIONS

Xclock accepts all of the standard X Toolkit command line options along with the additional options listed below:

–help This option indicates that a brief summary of the allowed options should be printed on the standard error.

–analog This option indicates that a conventional 12 hour clock face with tick marks and hands should be used. This is the default.

–digital This option indicates that a 24 hour digital clock should be used.

–chime This option indicates that the clock should chime once on the half hour and twice on the hour.

–hd *color*
This option specifies the color of the hands on an analog clock. The default is *black.*

–hl *color*
This option specifies the color of the edges of the hands on an analog clock, and is only useful on color displays. The default is *black.*

–update *seconds*
This option specifies the frequency in seconds at which *xclock* should update its display. If the clock is obscured and then exposed, it will be updated immediately. A value of less than 30 seconds will enable a second hand on an analog clock. The default is 60 seconds.

–padding *number*
This option specifies the width in pixels of the padding between the window border and clock text or picture. The default is 10 on a digital clock and 8 on an analog clock.

The following standard X Toolkit command line arguments are commonly used with *xclock:*

–bg *color*
: This option specifies the color to use for the background of the window. The default is *white.*

–bd *color*
: This option specifies the color to use for the border of the window. The default is *black.*

–bw *number*
: This option specifies the width in pixels of the border surrounding the window.

–fg *color*
: This option specifies the color to use for displaying text. The default is *black.*

–fn *font* This option specifies the font to be used for displaying normal text. The default is *6x10.*

–rv This option indicates that reverse video should be simulated by swapping the foreground and background colors.

–geometry *geometry*
: This option specifies the prefered size and position of the clock window.

–display *host:display*
: This option specifies the X server to contact.

–xrm *resourcestring*
: This option specifies a resource string to be used.

X DEFAULTS

This program uses the *Clock* widget in the X Toolkit. It understands all of the core resource names and classes as well as:

width (class **Width)**
: Specifies the width of the clock. The default for analog clocks is 164 pixels; the default for digital clocks is whatever is needed to hold the clock when displayed in the chosen font.

height (class **Height)**
: Specifies the height of the clock. The default for analog clocks is 164 pixels; the default for digital clocks is whatever is needed to hold the clock when displayed in the chosen font.

update (class **Interval**)
Specifies the frequency in seconds at which the time should be redisplayed.

foreground (class **Foreground**)
Specifies the color for the tic marks. The default is *black* since the core default for background is *white*.

hands (class **Foreground**)
Specifies the color of the insides of the clock's hands.

highlight (class **Foreground**)
Specifies the color used to highlight the clock's hands.

analog (class **Boolean**)
Specifies whether or not an analog clock should be used instead of a digital one. The default is True.

chime (class **Boolean**)
Specifies whether or not a bell should be rung on the hour and half hour.

padding (class **Margin**)
Specifies the amount of internal padding in pixels to be used. The default is 8.

font (class **Font**)
Specifies the font to be used for the digital clock. Note that variable width fonts currently will not always display correctly.

reverseVideo (class **ReverseVideo**)
Specifies that the foreground and background colors should be reversed.

SEE ALSO

X(1), xrdb(1), time(3C), Athena Clock widget

BUGS

Xclock believes the system clock.

When in digital mode, the string should be centered automatically.

When specifying a time offset, the grammar requires an hours field but if only minutes are given they will be quietly ignored. A negative offset of less than 1 hour is treated as a positive offset.

Digital clock windows default to the analog clock size.

Border color has to be explicitly specified when reverse video is used.

COPYRIGHT

AUTHORS

Tony Della Fera (MIT-Athena, DEC)
Dave Mankins (MIT-Athena, BBN)
Ed Moy (UC Berkeley)

NAME

MIT X Consortium

SYNOPSIS

X11R4 is brought to you by the MIT X Consortium.

DESCRIPTION

X Window System research began in the summer of 1984 at MIT as an informal joint undertaking between the Laboratory for Computer Science and Project Athena. Since that time, researchers and engineers at numerous other universities and companies have been involved in the design and implementation.

The MIT X Consortium was formed in January of 1988 to further the development of the X Window System. It is part of the Laboratory for Computer Science at MIT, and has as its major goal the promotion of cooperation within the computer industry in the creation of standard software interfaces at all layers in the X Window System environment. MIT's role is to provide the vendor-neutral architectural and administrative leadership required to make this work. The Consortium is financially self-supporting, with membership open to any organization. There are two categories of membership, Member (for large organizations) and Affiliate (for smaller organizations). A list of members as of the public release of X11R4 is given below.

The Director of the X Consortium acts as the ultimate architect for all X specifications and software. The activities of the Consortium are overseen by an MIT Steering Committee, which helps set policy and provides strategic guidance and review of the Consortium's activities. An Advisory Committee made up of member representatives meets regularly to review the Consortium's plans, assess its progress, and suggest future directions.

Most of the Consortium's activities take place via electronic mail, with meetings when required. As designs and specifications take shape, interest groups are formed from experts in the participating organizations. Typically a small multi-organization architecture team leads the design, with others acting as close observers and reviewers. Once a complete specification is produced, it may be submitted for formal technical review by the Consortium as a proposed standard. The standards process includes public review (outside the Consortium) and a demonstration of proof of concept, typically established with a public, portable implementation of the specification, (although other methods are possible on a case by case basis).

The MIT staff of the Consortium also maintains a software and documentation collection, containing both Consortium-developed and user-contributed materials, and endeavors to make periodic distributions of this collection available to the public without license and for a minimal fee. X11R4 is the fourth such distribution. The Consortium also sponsors an annual conference, open to the public, to promote the exchange of technical information about X.

STAFF

The Director of the X Consortium is Bob Scheifler, rws@expo.lcs.mit.edu. The MIT staff members are as follows.

Donna Converse, converse@expo.lcs.mit.edu
Jim Fulton, jim@expo.lcs.mit.edu
Michelle Leger, michelle@expo.lcs.mit.edu
Keith Packard, keith@expo.lcs.mit.edu
Chris Peterson, kit@expo.lcs.mit.edu

Ralph Swick, swick@athena.mit.edu, an employee of Digital Equipment Corporation working at MIT Project Athena, also works directly with the Consortium staff.

POSTAL ADDRESS

The MIT X Consortium can be reached by writing to

Bob Scheifler
MIT X Consortium
Laboratory for Computer Science
545 Technology Square
Cambridge, MA 02139

MEMBERS

Apple Computer, Inc.
American Telephone & Telegraph Company
Bull S.A.
Control Data Corporation
Cray Research, Inc.
Data General Corporation
Digital Equipment Corporation
Eastman Kodak Company
Fujitsu Limited
Hewlett-Packard Company
International Business Machines Corporation
Mitsubishi Electric Corporation
Motorola, Inc.
NEC Corporation
NCR Corporation
Nippon Telegraph and Telephone Corporation

OMRON Corporation
Prime Computer, Inc.
Rich, Inc.
Sequent Computer Systems Inc.
Siemens AG
Silicon Graphics, Inc.
Sony Corporation
Sun Microsystems, Inc.
Tektronix, Inc.
Texas Instruments, Inc.
Unisys Corporation
Wang Laboratories, Inc.
Xerox Corporation

AFFILIATES

ACER America Corporation
Adobe Systems
Advanced Graphics Engineering
Carnegie Mellon University
CETIA - Compagnie Europeene des Techniques de l'Ingenierie Assistee
Codonics, Inc.
Evans & Sutherland
GfxBase
INESC - Instituto de Engenharia de Sistemas e Computadores
Integrated Solutions
Interactive Systems Corporation
Interactive Development Environments
Integrated Computer Solutions, Inc.
Key Systems Engineering Corp.
Jupiter Systems
University of Kent at Canterbury
Landmark Graphics Corporation
Locus Computing
University of Lowell
Matrox International
Megatek Corporation
MIPS Computer Systems
MITRE Corporation
Network Computing Devices
Nova Graphics International
Open Software Foundation
O'Reilly & Associates, Inc.
PCS Computer Systeme GmbH

Ramtek Corporation
Samsung Software America
SGIP - Societe de Gestion et d'Informatique Publicis
Software Productivity Consortium
Solbourne Computer Inc.
Spectragraphics Corporation
Stanford University
Stardent Computer
Tatung Science and Technology
UNICAD, Inc.
Visual Technology Inc.
X/Open Company Ltd.

NAME

xcutsel - interchange between cut buffer and selection

SYNOPSIS

xcutsel [*-toolkitoption* ...] [-selection *selection*] [-cutbuffer *number*]

DESCRIPTION

The *xcutsel* program is used to copy the current selection into a cut buffer and to make a selection that contains the current contents of the cut buffer. It acts as a bridge between applications that don't support selections and those that do.

By default, *xcutsel* will use the selection named PRIMARY and the cut buffer CUT_BUFFER0. Either or both of these can be overridden by command line arguments or by resources.

An *xcutsel* window has the following buttons:

quit When this button is pressed, *xcutsel* exits. Any selections held by *xcutsel* are automatically released.

copy PRIMARY to 0
When this button is pressed, *xcutsel* copies the current selection into the cut buffer.

copy 0 to PRIMARY
When this button is pressed, *xcutsel* converts the current contents of the cut buffer into the selection.

The button labels reflect the selection and cutbuffer selected by command line options or through the resource database.

When the ``copy 0 to PRIMARY'' button is activated, the button will remain inverted as long as *xcutsel* remains the owner of the selection. This serves to remind you which client owns the current selection. Note that the value of the selection remains constant; if the cutbuffer is changed, you must again activate the copy button to retrieve the new value when desired.

OPTIONS

Xcutsel accepts all of the standard X Toolkit command line options as well as the following:

–selection *name*
This option specifies the name of the selection to use. The default is PRIMARY. The only supported abbreviations for this option are ``-select'', ``-sel'' and ``-s'', as the standard toolkit option ``-selectionTimeout'' has a similar name.

–cutbuffer *number*

This option specifies the cut buffer to use. The default is cut buffer 0.

X DEFAULTS

This program accepts all of the standard X Toolkit resource names and classes as well as:

selection (class **Selection)**

This resource specifies the name of the selection to use. The default is PRIMARY.

cutBuffer (class **CutBuffer)**

This resource specifies the number of the cut buffer to use. The default is 0.

WIDGET NAMES

The following instance names may be used when user configuration of the labels in them is desired:

sel-cut (class **Command)**

This is the ''copy SELECTION to BUFFER'' button.

cut-sel (class **Command)**

This is the ''copy BUFFER to SELECTION'' button.

quit (class **Command)**

This is the ''quit'' button.

SEE ALSO

X(1), xclipboard(1), xterm(1), text widget documentation, individual client documentation for how to make a selection.

BUGS

There is no way to change the name of the selection or the number of the cut buffer while the program is running.

COPYRIGHT

Copyright 1988, Massachusetts Institute of Technology
See *X(1)* for a full statement of rights and permissions.

AUTHOR

Ralph R. Swick, DEC/MIT Project Athena

NAME

xditview — display ditroff DVI files

SYNOPSIS

xditview [-*toolkitoption* ...] [-option ...]

DESCRIPTION

The *xditview* program displays ditroff output on an X display. It uses special font metrics which match the font set distributed with X11vR3, so it does not require access to the server machine for font loading.

OPTIONS

Xditview accepts all of the standard X Toolkit command line options along with the additional options listed below:

–help This option indicates that a brief summary of the allowed options should be printed.

–page This option specifies the page number of the document to be displayed.

–backingStore *backing-store-type*
Redisplay of the DVI window can take upto a second or so, this option causes the server to save the window contents so that when it is scrolled around the viewport, the window is painted from contents saved in backing store. *backing-store-type* can be one of **Always, WhenMapped** or **NotUseful.**

The following standard X Toolkit command line arguments are commonly used with *xditview:*

–bg *color*
This option specifies the color to use for the background of the window. The default is *white.*

–bd *color*
This option specifies the color to use for the border of the window. The default is *black.*

–bw *number*
This option specifies the width in pixels of the border surrounding the window.

–fg *color*
This option specifies the color to use for displaying text. The default is *black.*

–fn *font* This option specifies the font to be used for displaying widget text. The default is *fixed.*

−rv This option indicates that reverse video should be simulated by swapping the foreground and background colors.

−geometry *geometry*
This option specifies the prefered size and position of the window.

−display *host*:*display*
This option specifies the X server to contact.

−xrm *resourcestring*
This option specifies a resource string to be used.

X DEFAULTS

This program uses the *Dvi* widget in the X Toolkit. It understands all of the core resource names and classes as well as:

width (class **Width**)
Specifies the width of the window.

height (class **Height**)
Specifies the height of the window.

foreground (class **Foreground**)
Specifies the default foreground color.

font (class **Font**)
Specifies the font to be used for error messages.

USING XDITVIEW WITH DITROFF

To build a DVI file suitable for use with xditview, use the device description in devX75:

```
$ cd devX75
$ makedev DESC
$ mkdir /usr/lib/font/devX75
$ cp *.out /usr/lib/font/devX75
$ ditroff -TX75 ditroff-input | xditview
```

SEE ALSO

X(1), xrdb(1), ditroff(1)

BUGS

Xditview can be easily confused by attempting to display a DVI file constructed for the wrong device.

ORIGIN

Portions of this program originated in xtroff which was derived from suntroff.

COPYRIGHT

AUTHORS

Keith Packard (MIT X Consortium)
Richard L. Hyde (Purdue)
David Slattengren (Berkeley)
Malcom Slaney (Schlumberger Palo Alto Research)
Mark Moraes (University of Toronto)

NAME

xdm – X Display Manager

SYNOPSIS

xdm [-config *configuration_file*] [-daemon] [-debug *debug_level*] [-error *error_log_file*] [-nodaemon] [-resources *resource_file*] [-server *server_entry*] [-session *session_program*] [-xrm *resource_specification*]

DESCRIPTION

Xdm manages a collection of X displays, both local and possibly remote — the emergence of X terminals guided the design of several parts of this system, along with the development of the X Consortium standard XDMCP, the *X Display Manager Control Protocol*). It is designed to provide services similar to that provided by init, getty and login on character terminals: prompting for login/password, authenticating the user and running a ''session''.

A ''session'' is defined by the lifetime of a particular process; in the traditional character-based terminal world, it is the user's login shell process. In the *xdm* context, it is an arbitrary session manager. This is because in a windowing environment, a user's login shell process would not necessarily have any terminal-like interface with which to connect.

Until real session managers become widely available, the typical *xdm* substitute would be either a window manager with an exit option, or a terminal emulator running a shell - with the condition that the lifetime of the terminal emulator is the lifetime of the shell process that it is running - thus degenerating the X session to an emulation of the character-based terminal session.

When the session is terminated, *xdm* resets the X server and (optionally) restarts the whole process.

Because *xdm* provides the first interface that users will see, it is designed to be simple to use and easy to customize to the needs of a particular site. *Xdm* has many options, most of which have reasonable defaults. Browse through the various sections, picking and choosing the things you want to change. Pay particular attention to the **Xsession** section, which will describe how to set up the style of session desired.

OPTIONS

First, note that all of these options, except **-config**, specify values which can also be specified in the configuration file as resources.

-config *configuration_file*

Specifies a resource file which specifies the remaining configuration parameters. If no file is specified and the file */usr/lib/X11/xdm/xdm-config* exists, *xdm* will use it.

-daemon
Specifies "true" as the value for the **DisplayManager.daemonMode** resource. This makes *xdm* close all file descriptors, disassociate the controlling terminal and put itself in the background when it first starts up (just like the host of other daemons). It is the default behavior.

-debug *debug_level*
Specifies the numeric value for the **DisplayManager.debugLevel** resource. A non-zero value causes *xdm* to print piles of debugging statements to the terminal; it also disables the **DisplayManager.daemonMode** resource, forcing *xdm* to run synchronously. To interpret these debugging messages, a copy of the source code for xdm is almost a necessity. No attempt has been made to rationalize or standardize the output.

-error *error_log_file*
Specifies the value for the **DisplayManager.errorLogFile** resource. This file contains errors from *xdm* as well as anything written to stderr by the various scripts and programs run during the progress of the session.

-nodaemon
Specifies "false" as the value for the **DisplayManager.daemonMode** resource.

-resources *resource_file*
Specifies the value for the **DisplayManager*resources** resource. This file is loaded using *xrdb (1)* to specify configuration parameters for the authentication widget.

-server *server_entry*
Specifies the value for the **DisplayManager.servers** resource. See the section below which describes this resource in depth.

-udpPort *port_number*
Specifies the value for the **DisplayManager.requestPort** resource. This sets the port-number which XDM will monitor for XDMCP requests. As XDMCP uses the registered well-known udp port 177, this resource should probably not be changed except for debugging.

-session *session_program*
Specifies the value for the **DisplayManager*session** resource. This indicates the program to run when the user has logged in as the session.

-xrm *resource_specification*
: This allows an arbitrary resource to be specified, just as most toolkit applications.

RESOURCES

At many stages the actions of *xdm* can be controlled through the use of the configuration file, which is in the familiar X resource format. See Jim Fulton's article on resource files (*doc/tutorials/resources.txt*) for a description of the format. Some resources modify the behavior of *xdm* on all displays, while others modify its behavior on one single display. Where actions relate to a specific display, the display name is inserted into the resource name between ''DisplayManager'' and the final resource name segment. For example, **DisplayManager.expo_0.startup** is the name of the resource which defines the startup shell file on the ''expo:0'' display. Because the resource manager uses colons to separate the name of the resource from its value and dots to separate resource name parts, *xdm* substitutes underscores for the dots and colons when generating the resource name.

DisplayManager.servers
: This resource either specifies a file name full of server entries, one per line (if the value starts with a slash), or a single server entry. Each entry indicates a displays which should constantly be managed and which is not using XDMCP. Each entry consists of at least three parts: a display name, a display class, a display type, and (for local servers) a command line to start the server. A typical entry for local display number 0 would be:

```
:0 Digital-QV local /usr/bin/X11/X :0
```

The display types are:

local	a local display, i.e. one which has a server program to run
foreign	a remote display, i.e. one which has no server program to run

The display name must be something that can be passed in the **-display** option to any X program. This string is used in the display-specific resources to specify the particular display, so be careful to match the names (e.g. use ":0 local /usr/bin/X11/X :0" instead of "localhost:0 local /usr/bin/X11/X :0" if your other resources are specified as "DisplayManager._0.session"). The display class portion is also used in the display-specific resources, as the class portion of the resource. This is useful if you have a large collection of similar displays (like a corral of X terminals) and would like to set resources for groups of them. When using XDMCP, the display is required to specify the display class, so perhaps your X terminal documentation describes a reasonably standard display class string for your device.

DisplayManager.requestPort
This indicates the UDP port number which *xdm* uses to listen for incoming XDMCP requests. Unless you need to debug the system, leave this with it's default value of 177.

DisplayManager.errorLogFile
Error output is normally directed at the system console. To redirect it simply set this resource to any file name. A method to send these messages to syslog should be developed for systems which support it; however the wide variety of "standard" interfaces precludes any system-independent implementation. This file also contains any output directed to stderr by *Xstartup, Xsession and Xreset*, so it will contain descriptions of problems in those scripts as well.

DisplayManager.debugLevel
A non-zero value specified for this integer resource will enable reams of debugging information to be printed. It also disables daemon mode which would redirect the information into the bit-bucket. Specifying a non-zero debug level also allows non-root users to run *xdm* which would normally not be useful.

DisplayManager.daemonMode
Normally, *xdm* attempts to make itself into an unassociated daemon process. This is accomplished by forking and leaving the parent process to exit, then closing file descriptors and mangling the controlling terminal. When attempting to debug *xdm* , this is quite bothersome. Setting this resource to "false" will disable this feature.

DisplayManager.pidFile
The filename specified will be created to contain an ascii representation of the process-id of the main **xdm** process. This is quite useful when reinitializing the system. *Xdm* also uses file locking to attempt to eliminate multiple daemons running on the same machine, which would cause quite a bit of havoc.

DisplayManager.lockPidFile
This is the resource which controls whether *xdm* uses file locking to keep multiple xdms from running amok. On SYSV, this uses the lockf library call, while on BSD it uses flock. The default value is "true".

DisplayManager.remoteAuthDir

This is a directory name which *xdm* uses to temporarily store authorization files for displays using XDMCP. The default value is /usr/lib/X11/xdm.

DisplayManager.autoRescan"

This boolean controls whether *xdm* rescans the configuration file and servers file after a session terminates and the files have changed. By default it is "true". You can force *xdm* to reread these files by sending a SIGHUP to the main process.

DisplayManager.removeDomainname

When computing the display name for XDMCP clients, the resolver will typically create a fully qualified host name for the terminal. As this is sometimes confusing, *xdm* will remove the domain name portion of the host name if it is the same as the domain name for the local host when this variable is set. By default the value is "true".

DisplayManager.keyFile

XDM-AUTHENTICATION-1 style XDMCP authentication requires that a private key be shared between *xdm* and the terminal. This resource specifies the file containing those values. Each entry in the file consists of a display name and the shared key. By default, *xdm* does not include support for XDM-AUTHENTICATION-1 as it requires DES which is not generally distributable.

DisplayManager.DISPLAY.resources

This resource specifies the name of the file to be loaded by *xrdb (1)* as the resource data-base onto the root window of screen 0 of the display. This resource data base is loaded just before the authentication procedure is started, so it can control the appearance of the "login" window. See the section below on the authentication widget which describes the various resources which are appropriate to place in this file. There is no default value for this resource, but the conventional name is **/usr/lib/X11/xdm/Xresources.**

DisplayManager.DISPLAY.xrdb

Specifies the program used to load the resources. By default, *xdm* uses */usr/bin/X11/xrdb.*

DisplayManager.DISPLAY.cpp

This specifies the name of the C preprocessor which is used by xrdb.

DisplayManager.DISPLAY.startup

This specifies a program which is run (as root) after the authentication process succeeds. By default, no program is run. The conventional name for a file used here is *Xstartup.* See the **Xstartup** section below.

DisplayManager.DISPLAY.session

This specifies the session to be executed (not running as root). By default, */usr/bin/X11/xterm* is run. The conventional name is *Xsession.* See the **Xsession** session below.

DisplayManager.DISPLAY.reset

This specifies a program which is run (as root) after the session terminates. Again, by default no program is run. The conventional name is *Xreset.* See the **Xreset** section further on in this document.

DisplayManager.DISPLAY.openDelay

DisplayManager.DISPLAY.openRepeat

DisplayManager.DISPLAY.openTimeout

DisplayManager.DISPLAY.startAttempts

These numeric resources control the behavior of *xdm* when attempting to open intransigent servers. **openDelay** is the length of the pause (in seconds) between successive attempts, **openRepeat** is the number of attempts to make, **openTimeout** is the amount of time to wait while actually attempting the open (i.e. the maximum time spent in the *connect (2)* syscall) and **startAttempts** is the number of times this entire process is done before giving up on the server. After **openRepeat** attempts have been made, or if **openTimeout** seconds elapse in any particular attempt, *xdm* terminates and restarts the server, attempting to connect again, this process is repeated **startAttempts** time, at which point the display is declared dead and disabled. Although this behavior may seem arbitrary, it has been empirically developed and works quite well on most systems. The default values are 5 for **openDelay**, 5 for **openRepeat**, 30 for **openTimeout** and 4 for **startAttempts**.

DisplayManager.DISPLAY.pingInterval

DisplayManager.DISPLAY.pingTimeout

To discover when remote displays disappear, *xdm* occasionally "pings" them, using an X connection and sending XSync requests. **pingInterval** specifies the time (in minutes) between each ping attempt, **pingTimeout** specifies the maximum amount of time (in minutes) to wait for the terminal to respond to the request. If the terminal does not respond, the session is declared dead and terminated. By default, both are set to 5 minutes. *xdm* will not ping local displays. Although it would seem harmless, it is unpleasant when the workstation session is terminated as a result of the server hanging for NFS service and not responding to the ping.

DisplayManager.DISPLAY.terminateServer
This boolean resource specifies whether the X server should be terminated when a session terminates (instead of resetting it). This option can be used when the server tends to grow without bound over time in order to limit the amount of time the server is run. The default value is "FALSE".

DisplayManager.DISPLAY.userPath
Xdm sets the PATH environment variable for the session to this value. It should be a colon separated list of directories, see *sh(1)* for a full description. The default value can be specified in the X system configuration file with DefUserPath, frequently it is set to ":/bin:/usr/bin:/usr/bin/X11:/usr/ucb".

DisplayManager.DISPLAY.systemPath
Xdm sets the PATH environment variable for the startup and reset scripts to the value of this resource. The default for this resource is specified with the DefaultSystemPath entry in the system configuration file, but it is frequently "/etc:/bin:/usr/bin:/usr/bin/X11:/usr/ucb". Note the conspicuous absence of "." from this entry. This is a good practise to follow for root; it avoids many common trojan horse system penetration schemes.

DisplayManager.DISPLAY.systemShell
Xdm sets the SHELL environment variable for the startup and reset scripts to the value of this resource. By default, it is "/bin/sh".

DisplayManager.DISPLAY.failsafeClient
If the default session fails to execute, *xdm* will fall back to this program. This program is executed with no arguments, but executes using the same environment variables as the session would have had (see the section "Xsession" below). By default, */usr/bin/X11/xterm* is used.

DisplayManager.DISPLAY.grabServer

DisplayManager.DISPLAY.grabTimeout
To eliminate obvious security shortcomings in the X protocol, *xdm* grabs the server and keyboard while reading the name/password. The **grabServer** resource specifies if the server should be held for the duration of the name/password reading, when FALSE, the

server is ungrabbed after the keyboard grab succeeds, otherwise the server is grabbed until just before the session begins. The **grabTimeout** resource specifies the maximum time *xdm* will wait for the grab to succeed. The grab may fail if some other client has the server grabbed, or possibly if the network latencies are very high. This resource has a default value of 3 seconds; you should be cautious when raising it as a user can be spoofed by a look-alike window on the display. If the grab fails, *xdm* kills and restarts the server (if possible) and session.

DisplayManager.DISPLAY.authorize

DisplayManager.DISPLAY.authName

authorize is a boolean resource which controls whether *xdm* generates and uses authorization for the server connections. If authorization is used, **authName** specifies the type to use. Currently, xdm supports only MIT-MAGIC-COOKIE-1 authorization, XDM-AUTHORIZATION-1 could be supported as well, but DES is not generally distributable. XDMCP connections specify which authorization types are supported dynamically, so **authName** is ignored in this case. When **authorize** is set for a display and authorization is not available, the user is informed by having a different message displayed in the login widget. By default, **authorize** is "true"; **authName** is "MIT-MAGIC-COOKIE-1".

DisplayManager.DISPLAY.authFile"

This file is used to communicate the authorization data from **xdm** to the server, using the *-auth* server command line option. It should be kept in a directory which is not world-writable as it could easily be removed, disabling the authorization mechanism in the server.

DisplayManager.DISPLAY.resetForAuth

The original implementation of authorization in the sample server reread the authorization file at server reset time, instead of when checking the initial connection. As *xdm* generates the authorization information just before connecting to the display, an old server would not get up-to-date authorization information. This resource causes *xdm* to send SIGHUP to the server after setting up the file, causing an additional server reset to occur, during which time the new authorization information will be read

DisplayManager.DISPLAY.userAuthDir

When *xdm* is unable to write to the usual user authorization file ($HOME/.Xauthority), it creates a unique file name in this directory and points the environment variable XAUTHORITY at the created file. By default it uses "/tmp".

CONTROLLING THE SERVER

Xdm controls local servers using POSIX signals. SIGHUP is expected to reset the server, closing all client connections and performing other clean up duties. SIGTERM is expected to terminate the server. If these signals do not perform the expected actions, *xdm* will not perform properly.

To control remote servers not using XDMCP, *xdm* searches the window hierarchy on the display and uses the protocol request KillClient in an attempt to clean up the terminal for the next session. This may not actually kill all of the clients, as only those which have created windows will be noticed. XDMCP provides a more sure mechanism; when xdm closes it's initial connection, the session is over and the terminal is required to close all other connections.

CONTROLLING XDM

Xdm responds to two signals: SIGHUP and SIGTERM. When sent a SIGHUP, *xdm* rereads the configuration file and the 1file specified by the **DisplayManager.servers** resource and notices if entries have been added or removed. If a new entry has been added, *xdm* starts a session on the associated display. Entries which have been removed are disabled immediately, meaning that any session in progress will be terminated without notice, and no new session will be started.

When sent a SIGTERM, *xdm* terminates all sessions in progress and exits. This can be used when shutting down the system.

Xdm attempts to mark the various sub-processes for ps(1) by editing the command line argument list in place. Because xdm can't allocate additional space for this task, it is useful to start xdm with a reasonably long command line (15 to 20 characters should be enough). Each process which is servicing a display is marked "-<Display-Name>".

AUTHENTICATION WIDGET

The authentication widget is an application which reads a name/password pair from the keyboard. As this is a toolkit client, nearly every imaginable parameter can be controlled with a resource. Resources for this widget should be put into the file named by **DisplayManager.DISPLAY.resources**. All of these have reasonable default values, so it is not necessary to specify any of them.

xlogin.Login.width, xlogin.Login.height, xlogin.Login.x, xlogin.Login.y
The geometry of the login widget is normally computed automatically. If you wish to position it elsewhere, specify each of these resources.

xlogin.Login.foreground
The color used to display the typed-in user name.

xlogin.Login.font
The font used to display the typed-in user name.

xlogin.Login.greeting
A string which identifies this window. The default is "Welcome to the X Window System".

xlogin.Login.unsecureGreeting
When X authorization is requested in the configuration file for this display and none is in use, this greeting replaces the standard greeting. It's default value is "This is an unsecure session"

xlogin.Login.greetFont
The font used to display the greeting.

xlogin.Login.greetColor
The color used to display the greeting.

xlogin.Login.namePrompt
The string displayed to prompt for a user name. *Xrdb* strips trailing white space from resource values, so to add spaces at the end of the prompt (usually a nice thing), add spaces escaped with backslashes. The default is "Login: "

xlogin.Login.passwdPrompt
The string displayed to prompt for a password. The default is "Password: ".

xlogin.Login.promptFont
The font used to display both prompts.

xlogin.Login.promptColor
The color used to display both prompts.

xlogin.Login.fail
A message which is displayed when the authentication fails. The default is "Login Failed, please try again".

xlogin.Login.failFont
The font used to display the failure message.

xlogin.Login.failColor
The color used to display the failure message.

xlogin.Login.failTimeout
The time (in seconds) that the fail message is displayed. The default is 30 seconds.

xlogin.Login.translations

This specifies the translations used for the login widget. Refer to the X Toolkit documentation for a complete discussion on translations. The default translation table is:

```
Ctrl<Key>H:          delete-previous-character() \n\
Ctrl<Key>D:          delete-character() \n\
Ctrl<Key>B:          move-backward-character() \n\
Ctrl<Key>F:          move-forward-character() \n\
Ctrl<Key>A:          move-to-begining() \n\
Ctrl<Key>E:          move-to-end() \n\
Ctrl<Key>K:          erase-to-end-of-line() \n\
Ctrl<Key>U:          erase-line() \n\
Ctrl<Key>X:          erase-line() \n\
Ctrl<Key>C:          restart-session() \n\
Ctrl<Key>\\:         abort-session() \n\
<Key>BackSpace:      delete-previous-character() \n\
<Key>Delete:         delete-previous-character() \n\
<Key>Return:         finish-field() \n\
<Key>:               insert-char() \
```

The actions which are supported by the widget are:

delete-previous-character
: Erases the character before the cursor.

delete-character
: Erases the character after the cursor.

move-backward-character
: Moves the cursor backward.

move-forward-character
: Moves the cursor forward.

move-to-begining
: (Apologies about the spelling error.) Moves the cursor to the beginning of the editable text.

move-to-end
: Moves the cursor to the end of the editable text.

erase-to-end-of-line
: Erases all text after the cursor.

erase-line
: Erases the entire text.

finish-field
: If the cursor is in the name field, proceeds to the password field; if the cursor is in the password field, check the current name/password pair. If the name/password pair are valid, *xdm* starts the session. Otherwise the failure message is displayed and the user is prompted to try again.

abort-session
: Terminates and restarts the server.

abort-display
: Terminates the server, disabling it. This is a rash action and is not accessible in the default configuration. It can be used to stop *xdm* when shutting the system down, or when using xdmshell.

restart-session
: Resets the X server and starts a new session. This can be used when the resources have been changed and you want to test them, or when the screen has been overwritten with system messages.

insert-char
: Inserts the character typed.

set-session-argument
: Specifies a single word argument which is passed to the session at startup. See the sections on **Xsession** and **Typical usage**.

allow-all-access
: Disables access control in the server, this can be used when the .Xauthority file cannot be created by xdm. Be very careful using this, it might be better to disconnect the machine from the network before doing this.

The Xstartup file

This file is typically a shell script. It is run as "root" and should be very careful about security. This is the place to put commands which make fake entries in /etc/utmp, mount users' home directories from file servers, display the message of the day, or abort the session if logins are not allowed. Various environment variables are set for the use of this script:

DISPLAY	is set to the associated display name
HOME	is set to the home directory of the user
USER	is set to the user name
PATH	is set to the value of **DisplayManager.DISPLAY.systemPath**

SHELL	is set to the value of **DisplayManager.DISPLAY.systemShell**
XAUTHORITY	may be set to an authority file

No arguments of any kind are passed to the script. *Xdm* waits until this script exits before starting the user session. If the exit value of this script is non-zero, *xdm* discontinues the session immediately and starts another authentication cycle.

The Xsession program

This is the command which is run as the user's session. It is run with the permissions of the authorized user, and has several environment variables specified:

DISPLAY	is set to the associated display name
HOME	is set to the home directory of the user
USER	is set to the user name
PATH	is set to the value of **DisplayManager.DISPLAY.userPath**
SHELL	is set to the user's default shell (from /etc/passwd)
XAUTHORITY	may be set to a non-standard authority file

At most installations, *Xsession* should look in $HOME for a file *.xsession* which would contain commands that each user would like to use as a session. This would replace the system default session. *Xsession* should also implement the system default session if no user-specified session exists. See the section **Typical Usage** below.

An argument may be passed to this program from the authentication widget using the 'set-session-argument' action. This can be used to select different styles of session. One very good use of this feature is to allow the user to escape from the ordinary session when it fails. This would allow users to repair their own *.xsession* if it fails, without requiring administrative intervention. The section on typical usage demonstrates this feature.

The Xreset file

Symmetrical with *Xstartup*, this script is run after the user session has terminated. Run as root, it should probably contain commands that undo the effects of commands in *Xstartup*, removing fake entries from */etc/utmp* or unmounting directories from file servers. The collection of environment variables that were passed to *Xstartup* are also given to *Xreset*.

Typical Usage

Actually, *xdm* is designed to operate in such a wide variety of environments that "typical" is probably a misnomer. However, this section will focus on making *xdm* a superior solution to traditional means of starting X from /etc/ttys or manually.

First off, the *xdm* configuration file should be set up. A good thing to do is to make a directory (*/usr/lib/X11/xdm* comes immediately to mind) which will contain all of the relevant files. Here is a reasonable configuration file, which could be named *xdm-config* :

```
DisplayManager.servers:          /usr/lib/X11/xdm/Xservers
DisplayManager.errorLogFile:     /usr/lib/X11/xdm/xdm-errors
DisplayManager.pidFile:          /usr/lib/X11/xdm/xdm-pid
DisplayManager*resources:        /usr/lib/X11/xdm/Xresources
DisplayManager*session:          /usr/lib/X11/xdm/Xsession
DisplayManager._0.authorize:     true
DisplayManager*authorize:        false
```

As you can see, this file simply contains references to other files. Note that some of the resources are specified with "*" separating the components. These resources can be made unique for each different display, by replacing the "*" with the display-name, but normally this is not very useful. See the **Resources** section for a complete discussion.

The first file */usr/lib/X11/xdm/Xservers* contains the list of displays to manage. Most workstations have only one display, numbered 0, so the file will look like this:

```
        :0 Local local /usr/bin/X11/X :0
```

This will keep */usr/bin/X11/X* running on this display and manage a continuous cycle of sessions.

The file */usr/lib/X11/xdm/xdm-errors* will contain error messages from *xdm* and anything output to stderr by *Xstartup, Xsession or Xreset.* When you have trouble getting *xdm* working, check this file to see if *xdm* has any clues to the trouble.

The next configuration entry, */usr/lib/X11/xdm/Xresources*, is loaded onto the display as a resource database using *xrdb (1).* As the authentication widget reads this database before starting up, it usually contains parameters for that widget:

```
xlogin*login.translations: #override\
   <Key>F1: set-session-argument(failsafe) finish-field()\n\
   <Key>Return: set-session-argument() finish-field()
xlogin*borderWidth: 3
#ifdef COLOR
xlogin*greetColor: #f63
xlogin*failColor: red
xlogin*Foreground: black
```

```
xlogin*Background: #fdc
#else
xlogin*Foreground: black
xlogin*Background: white
#endif
```

The various colors specified here look reasonable on several of the displays we have, but may look awful on other monitors. As X does not currently have any standard color naming scheme, you might need to tune these entries to avoid disgusting results. Please note the translations entry; it specifies a few new translations for the widget which allow users to escape from the default session (and avoid troubles that may occur in it). Note that if #override is not specified, the default translations are removed and replaced by the new value, not a very useful result as some of the default translations are quite useful (like "<Key>: insert-char ()" which responds to normal typing).

The *Xstartup* file used here simply prevents login while the file */etc/nologin* exists. As there is no provision for displaying any messages here (there isn't any core X client which displays files), the user will probably be baffled by this behavior. I don't offer this as a complete example, but simply a demonstration of the available functionality.

Here is a sample *Xstartup* script:

```
#!/bin/sh
#
# Xstartup
#
# This program is run as root after the user is verified
#
if [ -f /etc/nologin ]; then
        exit 1
fi
exit 0
```

The most interesting script is *Xsession.* This version recognizes the special "failsafe" mode, specified in the translations in the *Xresources* file above, to provide an escape from the ordinary session:

```
#!/bin/sh
#
# Xsession
#
# This is the program that is run as the client
# for the display manager.  This example is
# quite friendly as it attempts to run a per-user
# .xsession file instead of forcing a particular
# session layout
#

case $# in
1)
        case $1 in
        failsafe)
                exec xterm -geometry 80x24-0-0 -ls
                ;;
        esac
esac

startup=$HOME/.xsession
resources=$HOME/.Xresources

if [ -f $startup ]; then
        exec $startup
        exec /bin/sh $startup
else
        if [ -f $resources ]; then
                xrdb -load $resources
        fi
        twm &
        exec xterm -geometry 80x24+10+10 -ls
fi
```

No *Xreset* script is necessary, so none is provided.

SOME OTHER POSSIBILITIES

You can also use *xdm* to run a single session at a time, using the 4.3 *init* options or other suitable daemon by specifying the server on the command line:

```
xdm -server ":0 SUN-3/60CG4 local /usr/bin/X :0"
```

Or, you might have a file server and a collection of X terminals. The configuration for this could look identical to the sample above, except the *Xservers* file might look like:

```
extol:0 VISUAL-19 foreign
exalt:0 NCD-19 foreign
explode:0 NCR-TOWERVIEW3000 foreign
```

This would direct *xdm* to manage sessions on all three of these terminals. See the section "Controlling Xdm" above for a description of using signals to enable and disable these terminals in a manner reminiscent of init(8).

One thing that *xdm* isn't very good at doing is coexisting with other window systems. To use multiple window systems on the same hardware, you'll probably be more interested in *xinit* .

SEE ALSO

X(1), xinit(1) and XDMCP

BUGS

COPYRIGHT

Copyright 1988, Massachusetts Institute of Technology.
See *X(1)* for a full statement of rights and permissions.

AUTHOR

Keith Packard, MIT X Consortium

NAME

xdpr – dump an X window directly to a printer

SYNOPSIS

xdpr [*filename*] [**–display** *host:display*] [**–P***printer*] [**–device** *printer_device*] [option ...]

DESCRIPTION

Xdpr uses the commands *xwd*(1), *xpr*(1), and *lpr*(1) to dump an X window, process it for a particular printer type, and print it out on the printer of your choice. This is the easiest way to get a printout of a window. *Xdpr* by default will print the largest possible representation of the window on the output page.

The options for *xdpr* are the same as those for *xpr*, *xwd*, and *lpr*. The most commonly-used options are described below; see the manual pages for these commands for more detailed descriptions of the many options available.

filename
Specifies a file containing a window dump (created by *xwd*) to be printed instead of selecting an X window.

-P*printer*
Specifies a printer to send the output to. If a printer name is not specified here, *xdpr* (really, *lpr*) will send your output to the printer specified by the *PRINTER* environment variable. Be sure that type of the printer matches the type specified with the *–device* option.

-display *host:display*[*.screen*]
Normally, *xdpr* gets the host and display number to use from the environment variable ''DISPLAY''. One can, however, specify them explicitly; see *X*(1).

-device *printer-device*
Specifies the device type of the printer. Available printer devices are "ln03" for the DEC LN03, "pp" for the IBM 3812 PagePrinter, and "ps" for any postscript printer (e.g. DEC LN03R or LPS40). The default is "ln03".

-help This option displays the list of options known to xdpr.

Any other arguments will be passed to the *xwd*(1), *xpr*(1), and *lpr*(1) commands as appropriate for each.

SEE ALSO

xwd(1), xpr(1), lpr(1), xwud(1), X(1)

ENVIRONMENT

DISPLAY - for which display to use by default.
PRINTER - for which printer to use by default.

COPYRIGHT

AUTHOR

Paul Boutin, MIT Project Athena
Michael R. Gretzinger, MIT Project Athena
Jim Gettys, MIT Project Athena

NAME

xdpyinfo - display information utility for X

SYNOPSIS

xdpyinfo [-display *displayname*]

DESCRIPTION

Xdpyinfo is a utility for displaying information about an X server. It is used to examine the capabilities of a server, the predefined values for various parameters used in communicating between clients and the server, and the different types of screens and visuals that are available.

EXAMPLE

The following shows a sample produced by *xdpyinfo* when connected to display that supports an 8 plane Pseudocolor screen as well as a 1 plane (monochrome) screen.

```
name of display:    empire:0.0
version number:    11.0
vendor string:    MIT X Consortium
vendor release number:    3
maximum request size:  16384 longwords (65536 bytes)
motion buffer size:  0
bitmap unit, bit order, padding:    32, MSBFirst, 32
image byte order:    MSBFirst
number of supported pixmap formats:    2
supported pixmap formats:
   depth 1, bits_per_pixel 1, scanline_pad 32
   depth 8, bits_per_pixel 8, scanline_pad 32
keycode range:    minimum 8, maximum 129
default screen number:    0
number of screens:    2

screen #0:
  dimensions:    1152x900 pixels (325x254 millimeters)
  resolution:    90x90 dots per inch
  root window id:    0x8006d
  depth of root window:    1 plane
  number of colormaps:    minimum 1, maximum 1
  default colormap:    0x80065
  default number of colormap cells:    2
  preallocated pixels:    black 1, white 0
  options:    backing-store YES, save-unders YES
  current input event mask:    0x1b8003c
    ButtonPressMask          ButtonReleaseMask        EnterWindowMask
    LeaveWindowMask          SubstructureNotifyMask   SubstructureRedirectMask
    FocusChangeMask        ColormapChangeMask      OwnerGrabButtonMask
  number of visuals:   1
  default visual id:  0x80064
  visual:
    visual id:   0x80064
    class:   StaticGray
    depth:   1 plane
    size of colormap:   2 entries
    red, green, blue masks:   0x0, 0x0, 0x0
    significant bits in color specification:   1 bits

screen #1:
  dimensions:   1152x900 pixels (325x254 millimeters)
  resolution:   90x90 dots per inch
  root window id:   0x80070
  depth of root window:   8 planes
  number of colormaps:   minimum 1, maximum 1
  default colormap:   0x80067
  default number of colormap cells:   256
  preallocated pixels:   black 1, white 0
  options:   backing-store YES, save-unders YES
  current input event mask:   0x0
  number of visuals:   1
  default visual id:  0x80066
  visual:
    visual id:   0x80066
    class:   PseudoColor
    depth:   8 planes
    size of colormap:   256 entries
    red, green, blue masks:   0x0, 0x0, 0x0
    significant bits in color specification:   8 bits
```

ENVIRONMENT

DISPLAY

To get the default host, display number, and screen.

SEE ALSO

X(1), xwininfo(1), xprop(1), xrdb(1)

BUGS

Due to a bug in the Xlib interface, there is currently no portable way to determine the depths of pixmap images that are supported by the server.

COPYRIGHT

AUTHOR

Jim Fulton, MIT X Consortium

NAME

xedit - simple text editor for X

SYNTAX

xedit [*-toolkitoption* ...] [filename]

OPTIONS

Xedit accepts all of the standard X Toolkit command line options (see *X(1)*), plus:

filename	Specifies the file that is to be loaded during start-up. This is the file which will be edited. If a file is not specified, *xedit* lets you load a file or create a new file after it has started up.

DESCRIPTION

Xedit provides a window consisting of the following three areas:

Commands Menu	Lists editing commands (for example, **Undo** or **Search**).
Message Window	Displays *xedit* messages. In addition, this window can be used as a scratch pad.
Edit Window	Displays the text of the file that you are editing or creating.

COMMANDS

Quit	Quits the current editing session. If any changes have not been saved, *xedit* displays a warning message and allows you to save the file.
Save	Stores a copy of the original, unedited file in *file*.BAK. Then, overwrites the original file with the edited contents.
Edit	Allows the text displayed in the Edit window to be edited.
Load	Loads the specified file and displays it in the Edit window.
Undo	Undoes the last edit only.
More	Undoes each edit previous to the last edit, which must first be undone with the **Undo** command.
Jump	Advances the cursor from the beginning of the file to the text line that corresponds to the selected line number.

<<	Searches from the cursor back to the beginning of the file for the string entered in the Search input box. If you do not enter a string in the Search input box, *xedit* automatically copies the last string that you selected from any X application into the Search input box and searches for that string.
Search >>	Searches from the cursor forward to the end of the file for the string entered in the search input box. If you do not enter a string in the Search input box, *xedit* automatically copies the last string that you selected from any X application into the Search input box and searches for that string.
Replace	Replaces the last searched-for string with the string specified in the Replace input box. If no string has been previously searched for, searches from the insert cursor to the end of the file for the next occurrence of the search string and highlights it.
All	Repositions the cursor at the beginning of the file and replaces all occurrences of the search string with the string specified in the Replace input box.

X DEFAULTS

For *xedit,* the available class identifiers are:

ButtonBox
Command
Scrollbar
Text

For *xedit,* the available name identifiers are:

All
Edit
EditWindow
Jump
Load
MessageWindow
More
Quit
Replace

Save
Undo
xedit

For *xedit*, the available resources are:

EnableBackups	Specifies that, when edits made to an existing file are saved, *xedit* is to copy the original version of that file to *file*.BAK before it saves the changes. If the value of this option is specified as off, a backup file is not created.
background	Specifies the background color to be displayed in command buttons. The default is white.
border	Specifies the border color of the *xedit* window.
borderWidth	Specifies the border width, in pixels, of the *xedit* window.
font	Specifies the font displayed in the *xedit* window.
foreground	Specifies the foreground color of the *xedit* window. The default is black.
geometry	Specifies the geometry (window size and screen location) to be used as the default for the *xedit* window. For information about the format of the geometry specification, see *X(1)*.
internalHeight	Specifies the internal horizontal padding (spacing between text and button border) for command buttons.
internalWidth	Specifies the internal vertical padding (spacing between text and button border) for command buttons.

KEY BINDINGS

Each specification included in the *XtActions* file modifies a key setting for the editor that *xedit* uses. When defining key specifications, you must use the following resource specification: text.EventBindings: .XtActions

Each key specification assigns an editor command to a named key and/or mouse combination and has the format:

key: *function*

key	Specifies the key or mouse button that is used to invoke the named function.
function	Specifies the function to be invoked when the named key is pressed.

For more information about specifications in the *XtActions* file, see *X(1)*.

FILES

~/.XtActions
/usr/lib/X11/.XtActions

SEE ALSO

X(1), xrdb(1)

RESTRICTIONS

Large numbers of certain edit functions (for example, Undo or More) tend to degrade performance over time. If there is a noticeable decrease in response time, save and reload the file.

BUGS

It is not clear how to select a line number for the *Jump* command.

The string searches don't work properly.

COPYRIGHT

Copyright 1988, Digital Equipment Corporation.

NAME

xev - print contents of X events

SYNOPSIS

xev [–display *displayname*] [–geometry *geom*]

DESCRIPTION

Xev creates a window and then asks the X server to send it notices called *events* whenever anything happens to the window (such as being moved, resized, typed in, clicked in, etc.). It is useful for seeing what causes events to occur and to display the information that they contain.

OPTIONS

–display *display*
This option specifies the X server to contact.

–geometry *geom*
This option specifies the size and/or location of the window.

SEE ALSO

X(1), xwininfo(1), xdpyinfo(1), Xlib Programmers Manual, X Protocol Specification

COPYRIGHT

Copyright 1988, Massachusetts Institute of Technology.
See *X(1)* for a full statement of rights and permissions.

AUTHOR

Jim Fulton, MIT X Consortium

NAME

xeyes – watch over your shoulder

SYNOPSIS

xeyes [-option ...]

DESCRIPTION

Xeyes watches what you do and reports to the Boss.

OPTIONS

–fg *foreground color*
: choose a different color for the pupil of the eyes.

–bg *background color*
: choose a different color for the background.

–outline *outline color*
: choose a different color for the outline of the eyes.

–center *center color*
: choose a different color for the center of the eyes.

–backing *{ WhenMapped Always NotUseful }*
: selects an appropriate level of backing store.

–geometry *geometry*
: define the initial window geometry; see *X(1)*.

–display *display*
: specify the display to use; see *X(1)*.

–bd *border color*
: choose a different color for the window border.

–bw *border width*
: choose a different width for the window border.

–shape uses the SHAPE extension to shape the window.

SEE ALSO

X(1), X Toolkit documentation

COPYRIGHT

Copyright 1988, Massachusetts Institute of Technology.
See *X(1)* for a full statement of rights and permissions.

AUTHOR

Keith Packard, MIT X Consortium

NAME

xfd - font displayer for X

SYNOPSIS

xfd [-options ...] -fn *fontname*

OPTIONS

Xfd accepts all of the standard toolkit command line options along with the additional options listed below:

–fn *font* This option specifies the font to be displayed.

–box This option indicates that a box outlining the area that would be filled with background color by an ImageText request.

–center This option indicates that each glyph should be centered in its grid.

–start *number*
This option specifies the glyph index of the upper left hand corner of the grid. This is used to view characters at arbitrary locations in the font. The default is 0.

–bc *color*
This option specifies the color to be used if ImageText boxes are drawn.

DESCRIPTION

The *xfd* utility creates a window containing the name of the font being displayed, a row of command buttons, several lines of text for displaying character metrics, and a grid containing one glyph per cell. The characters are shown in increasing order from left to right, top to bottom. The first character displayed at the top left will be character number 0 unless the -start option has been supplied in which case the character with the number given in the -start option will be used.

The characters are displayed in a grid of boxes, each large enough to hold any single character in the font. Each character glyph is drawn using the PolyText16 request (used by the *Xlib* routine **XDrawString16**). If the *-box* option is given, a rectangle will be drawn around each character, showing where an ImageText16 request (used by the *Xlib* routine **XDrawImage-String16**) would cause background color to be displayed.

The origin of each glyph is normally set so that the character is drawn in the upper left hand corner of the grid cell. However, if a glyph has a negative left bearing or an unusually large ascent, descent, or right bearing (as is the case with *cursor* font), some character may not appear in their own grid cells. The *-center* option may be used to force all glyphs to be centered in their respective cells.

All the characters in the font may not fit in the window at once. To see the next page of glyphs, press the *Next* button at the top of the window. To see the previous page, press *Prev*. To exit *xfd*, press *Quit*.

Individual character metrics (index, width, bearings, ascent and descent) can be displayed at the top of the window by pressing on the desired character.

The font name displayed at the top of the window is the full name of the font, as determined by the server. See *xlsfonts* for ways to generate lists of fonts, as well as more detailed summaries of their metrics and properties.

X DEFAULTS

To be written.

SEE ALSO

X(1), xlsfonts(1), xrdb(1)

BUGS

The program should skip over pages full of non-existent characters.

COPYRIGHT

AUTHOR

Jim Fulton, MIT X Consortium; previous program of the same name by Mark Lillibridge, MIT Project Athena.

NAME

xfig – Facility for Interactive Generation of figures under X11

SYNOPSIS

xfig [**-ri[ght]**] [**-le[ft]**] [**-L[andscape]**] [**-P[ortrait]**] [**-w[idth]** *inches*] [**-h[eight]** *inches*] [**-no[track]**] [**-tr[ack]**] [*file*]

DESCRIPTION

Xfig is a menu-driven tool that allows the user to draw and manipulate objects interactively in an X window. It runs under X version 11 release 2 and requires a three-button mouse. *File* specifies the name of a file to be edited. The description of objects in the file will be read at the start of *xfig*.

The output from *xfig* can be printed in several ways

troff - f2p (*xfig* to *pic*(1) translator, also known by its previous name *ftop*(1L)) is used to translate *xfig* files into *pic*(1) language. The resulting file may then be processed in the same maner as any other *pic* file.

postscript - f2ps (*xfig* to *postscript* translator) is used to produce a *postscript* file from an *xfig* file. The *postscript* file can be sent directly to a postscript printer.

LaTeX - fig2latex (*xfig* to *LaTeX* translator) produces a *LaTeX* file from an *xfig* file. This file contains *LaTeX* picture environment commands and can be processed along with other *LaTeX* commands.

PiCTeX - fig2tex (*xfig* to *PiCTeX* translator) produces a *PiCTeX* file from an *xfig* file. This file contains macros that can be used with the *PiCTeX* environment under *TeX* or *LaTeX*.

OPTIONS

–ri Change the position of the panel window to the right of the canvas window (default: left).

–le Change the position of the panel window to the left of the canvas window.

–L Make *xfig* come up in landscape mode (10" x 7.5").

–P Make *xfig* come up in portrait mode (7.5" x 10"). This is the default.

-w *inches*
Make *xfig* come up *inches* wide.

-h *inches*
Make *xfig* come up *inches* high.

-tr Turn on cursor (mouse) tracking arrows.

-no Turn off cursor (mouse) tracking arrows.

GRAPHICAL OBJECTS

The objects in *xfig* are divided into **primitive objects** and **compound object**. The primitive objects are: *ARC*, *CIRCLE*, *CLOSED SPLINE*, *ELLIPSE*, *POLYLINE*, *POLYGON*, *SPLINE*, and *TEXT*. A primitive can be moved, rotated, flipped, copied or erased. A compound object is composed of primitive objects. The primitive objects that constitute a compound can not be individually modified, but they can be manipulated as an entity; a compound can be moved, rotated, flipped, copied or erased. An extra function that can be applied to a compound object is **scaling**, which is not available for primitive objects.

DISPLAY WINDOWS

Five windows comprise the display area of *xfig*: the top ruler, the side ruler, the panel window, the message window, and the canvas window. The message window always appears below the others; it is the area in which messages are sent and received. The panel window can be placed to the left or right of the the canvas window (default: left).

POP-UP MENU

The pop-up menu appears when the right mouse button is pressed with the cursor positioned within the canvas window. Positioning the cursor over the desired menu entry and releasing the button selects a menu entry.

There are a number of file accessing functions in the pop-up menu. Most of the time when one of these functions is selected, the user will be asked for a file name. If the specified file can be located and the access permission are granted, *xfig* will carry out the function. However in case things go wrong, *xfig* will abort the function and printed the causes on the message window.

Undo Undo the last object creation or modification.

Redisplay
Redraw the canvas.

Remove all
Remove all objects on the canvas window (can be undone).

Edit file ...
The current contents of the canvas are cleared and objects are read from the specified file. The user will be asked for a file name. This file will become the current file.

Save Save the current contents of the canvas in the current file. If no file is being edited, the user will be asked for a file name as in the "Save in ..." function.

Read file ...
Read objects from the specified file and merge them with objects already shown on the canvas. (The user will be asked for a file name.)

Save as ...
Save objects on the screen into a file specified by the user. (The user will be asked for a file name.)

Status Show the name of the current file and directory.

Change Directory
Change the working directory. Any file name without a full path name will employ the current working directory.

Save & Exit
Save the objects in the current file and exit from *xfig*. If there is no current file, the user will be asked for a file name. No confirmation will be asked.

Quit Exit from *xfig*, discarding all objects. The user will be asked to confirm the action, by clicking the left button.

Save as BITMAP ...
Create a bitmap picture of the drawings for use with other tools (for example, for use as an icon). The smallest rectangular area of pixels that encompasses the figure is written to the named file (the user will be asked for a file name) from top row to bottom and left to right (in Sun raster format). Only *TEXT* objects that are parts of compound objects will be treated as parts of the picture; other texts are saved as objects in *xfig* format following the bitmap data. The coordinates of these text objects can be used to identify locations on the bitmap.

PANEL WINDOW MANIPULATION FUNCTIONS

Icons in the panel window represent object manipulation functions, modes and other drawing or modification aids. Manipulation functions are selected by positioning the cursor over it and clicking the left mouse button. The selected icon is highlighted, and a message describing its function appears in the message window.

The left and middle buttons are used to creat and modify objects in the canvas window. Most actions start with clicking of the left button and end with clicking of the right button. There is no need to hold down a button while positioning the cursor.

PANEL WINDOW COMMAND DESCRIPTIONS

Entries in the panel window can be classified into two categories: object creation/modification/removal commands (only one of which may be active at any one time), and drawing aids (which act as toggle switches). There are two ways for drawing circles, two for ellipses, two for splines and two for closed splines. There are two basic splines. One is the interpolated spline in which the spline pass thorough the entered points (knots). The other is the normal spline in which on control points are passed by the spline (except for the two end points in the open spline).

OBJECT CREATION/MODIFICATION/REMOVAL

Multiple commands are grouped thematically in the following descriptions (which is listed alphabetically).

ADD/DELETE ARROWS
: Add or delete arrow heads for *POLYLINE*, *POLYGON*, *SPLINE* or *CLOSED SPLINE* objects (points of a *BOX* can not be added or deleted).

ADD/DELETE POINTS
: Add or delete points for *POLYLINE*, *POLYGON*, *SPLINE* or *CLOSED SPLINE* objects (points of a *BOX* can not be added or deleted).

ARC Create an arc. Specify three points using the left button.

BOX Create rectangular boxes. Start with the left button and terminate with the right button.

BREAK COMPOUND
: Break a compound object to allow manipulation of its component parts. Click the left button on the bounding box of the compound object.

CIRCLE
: Create circles by specifying their radii or diameters. Click the left button on the canvas window, move the cursor until the desired radius or diameter is reached, then click the middle button to terminate. The circle will be drawn after the pressing of the middle button.

CLOSED INTERPOLATED SPLINE
: Create closed or periodic splines. The function is similar to *POLYGON* except that a closed interpolated spline is drawn. The spline will pass through the points (knots).

CLOSED SPLINE
: Create closed or periodic spline objects. The function is similar to *POLYGON* except that a closed spline will be drawn instead of polygon. The entered points are just control points; i.e., the spline will not pass any of these points.

COPY
: Copy object. Click the left button over part of the object to be copied (for *CIRCLE* and *ELLIPSE* objects, position on their circumferences). Drag the object to the desired position and click the middle button. This function as well as the following three functions (*MOVE, MOVE POINT, REMOVE*) will cause point markers (manipulation aids) to be shown on the canvas window. There are no markers for *CIRCLE* or *ELLIPSE* objects.

ELLIPSE
: Create ellipses using the same procedure as for the drawing of circles.

GLUE
: Glue the objects within a bounding box into a compound object (the bounding box itself is not part of the figure; it is a visual aid for manipulating the compound).

INTERPOLATED SPLINE
: Create (cubic spline) spline objects. Enter control vectors in the same way as for creation of a *POLYLINE* object. At least three points (two control vectors) must be entered. The spline will pass through the entered points.

MOVE
: Move objects in the same way as in *COPY.*

MOVE POINT
: Modify the position of points of *POLYLINE, BOX, POLYGON, ELLIPSE, ARC and SPLINE objects. Click the left button over the desired point, reposition the point, and click the middle button. Note that BOX and POLYGON objects are internally stored as POLYLINE objects, and therefore moving certain points may open these objects.*

POLYGON
: Same as *POLYLINE* except that a line segment is drawn connecting the first and last points entered.

POLYLINE
: Create polylines (line segments connecting a sequence of points). Enter points by clicking the left button at the desired positions on the canvas window. Click the middle button to terminate.

REMOVE
Remove (or delete) objects.

SCALE COMPOUND
Only compound objects can be scaled. Click the left button on a corner of the bounding box, stretch the bounding box to the desired size, and click the middle button. Or click the left button on a side of the bounding box, stretch that side to the desired size, and click the middle button.

SPLINE Create (quadratic spline) spline objects. Enter control vectors in the same way as for creation of a *POLYLINE* object. At least three points (two control vectors) must be entered. The spline will pass only the two end points.

TEXT Create text strings. Click the left button at the desired position on the canvas window, then enter text from the keyboard. A DEL or ^H (backspace) will delete a character, while a ^U will kill the entire line. Terminate by clicking the middle button or typing the return key. To edit text, click on an existing text string with the left button. Insertion of characters will take place at that point.

TURN Turn *POLYGON* into a *CLOSED INTERPOLATED SPLINE* object, or turn *POLYLINE* into a *INTERPOLATED SPLINE* object.

DRAWING AIDS

Drawing aids act as toggle switches. More than one can be selected at a time (except for *GRID* and the line drawing modes).

AUTO FORWARD/BACKWARD ARROW
Automatically add forward/backward arrow heads to *POLYLINE*, *SPLINE* or *ARC* objects.

FLIP Invert the object (middle button) or produce a mirror-image copy of an object (left button). Point to part of the object ("the handle"), click the appropriate button.

GRID Display either the quarter- or half-inch grids (left button).

MAGNET
Round points to the nearest 1/16 of an inch. This affects every function, and is provided as an alignment aid.

UNRESTRICTED
Allow lines to be drawn with any slope.

MANHATTAN
Enforce drawing of lines in the horizontal and vertical direction only. Both *MANHATTAN* and *MOUNTAIN* can be turned on simultaneously. The creations of *POLYGON*, *POLYLINE* and

SPLINE objects are affected by these two modes.

MOUNTAIN

Enforce drawing of only diagonal lines. Both *MANHATTAN* and *MOUNTAIN* can be turned on simultaneously. The creations of *POLYGON*, *POLYLINE* and *SPLINE* objects are affected by these two modes.

MANHATTAN MOUNTAIN

Allow lines to be drawn at any slope allowed when in MOUNTIAIN or MANHATTAN modes.

LATEX LINE

Allow lines to be drawn only at slopes which can be handled by LaTeX picture environment lines: slope = x/y, where x,y are integers in the range [-6,6].

LATEX VECTOR

Allow lines to be drawn only at slopes which can be handled by LaTeX picture environment vectors: slope = x/y, where x,y are integers in the range [-4,4].

ROTATE

Rotate the object (middle button) or copy (left button) +90 degrees.

SOLID/DASHED LINE STYLE

Toggle between solid and dashed line styles. The dash length is fixed at 0.05 inch.

X DEFAULTS

The overall widget name(Class) is xfig.fig(Fig.TopLevelShell). This set of resources correspond to the command line arguments:

trackCursor	(boolean:on) -track and -notrack arguments
justify	(boolean:false) -right and -left arguments
landscape	(boolean:false) -Landscape and -Portrait arguments
debug	(boolean:off) -debug arguments
width	(integer:7.5 or 10 inches) -width argument
height	(integer:10 or 7.5 inches) -height argument
reverseVideo	(boolean:off) -inverse argument

These arguments correspond to the widgets which make up *xfig*.

overall window	form(Form)
side panel	form.panel(Form.Box)
icons	form.panel.button(Form.Box.Command)
top ruler	form.truler(Form.Label)
side ruler	form.sruler(Form.Label)
canvas	form.canvas(Form.Label)
message window	form.message(Form.Command)
menu	form.popup_menu.menu(Form.OverrideShell.Box)
menu title	form.popup_menu.menu.title(Form.OverrideShell.Box.Label)
menu items	form.popup_menu.menu.pane(Form.OverrideShell.Box.Command)

For example, to set the background of the panel to blue the resource would be:
xfig*form.panel.background: blue

NOTE: The font used in the canvas cannot be changed at this time.

BUGS

Text strings will appear differently on hard copy, because the display fonts are fixed-width fonts while the fonts used by the typesetter systems are variable-width fonts.

A double quote in a text string should be preceded by a back slash if the it is to be printed through *pic*(1).

Objects that extend beyond the canvas window may cause image shrinkage in hard copy printed by *pic*(1), since it will try to fit every object onto a single 8.5" x 11" page.

Ellipses which are too narrow may cause *xfig* to loop forever.

Objects which are created while one of the *grids* is on may appear ragged. This can be corrected by selecting *Redisplay* from the pop-up menu.

The X11 cursors are not the original ones but chosen from X11's cursor font.

SEE ALSO

Brian W. Kernighan *PIC - A Graphics Language for Typesetting User Manual*
ditroff(1), f2p(1), f2ps(1), fig2latex(1), fig2tex(1), pic(1), troff(1), tex(1), latex(1)

ACKNOWLEDGEMENT

Many thanks goes to Professor Donald E. Fussell who inspired the creation of this tool.

AUTHORS

Original author:
Supoj Sutanthavibul
University of Texas at Austin
(supoj@sally.utexas.edu)

Manual page modified by:
R. P. C. Rodgers
UCSF School of Pharmacy
San Francisco, CA 94118

The LaTeX line drawing modes were contributed by:
Frank Schmuck
Cornell University

X11 port by:
Ken Yap
Rochester
(ken@cs.rochester.edu)

Variable window sizes, cleanup of X11 port, right hand side panel under X11, X11 manual page provided by:
Dana Chee
Bellcore
(dana@bellcore.com)

Cleanup of color port to X11 by:
John T. Kohl
MIT
(jtkohl@athena.mit.edu)

NAME

xfontsel - point & click interface for selecting X11 font names

SYNTAX

xfontsel [-*toolkitoption* ...] [**-pattern** *fontname*] [**-print**] [**-sample** *text*]

DESCRIPTION

The *xfontsel* application provides a simple way to display the fonts known to your X server, examine samples of each, and retrieve the X Logical Font Description ("XLFD") full name for a font.

If **-pattern** is not specified, all fonts with XLFD 14-part names will be selectable. To work with only a subset of the fonts, specify **-pattern** followed by a partially or fully qualified font name; e.g., ‘‘-pattern *medium*’’ will select that subset of fonts which contain the string ‘‘medium’’ somewhere in their font name. Be careful about escaping wildcard characters in your shell.

If **-print** is specified on the command line the selected font specifier will be written to standard output when the *quit* button is activated. Regardless of whether or not **-print** was specified, the font specifier may be made the PRIMARY (text) selection by activating the *select* button.

The **-sample** option specifies the sample text to be used to display the selected font, overriding the default.

INTERACTIONS

Clicking any pointer button in one of the XLFD field names will pop up a menu of the currently-known possibilities for that field. If previous choices of other fields were made, only values for fonts which matched the previously selected fields will be selectable; to make other values selectable, you must deselect some other field(s) by choosing the ‘‘*’’ entry in that field. Unselectable values may be omitted from the menu entirely as a configuration option; see the **ShowUnselectable** resource, below. Whenever any change is made to a field value, *xfontsel* will assert ownership of the PRIMARY_FONT selection. Other applications (see, e.g., *xterm*) may then retrieve the selected font specification.

Clicking the left pointer button in the *select* widget will cause the currently selected font name to become the PRIMARY text selection as well as the PRIMARY_FONT selection. This then allows you to paste the string into other applications. The **select** button remains highlighted to remind you of this fact, and de-highlights when some other application takes the PRIMARY selection away. The *select* widget is a toggle; pressing it when it is highlighted will cause *xfontsel* to release the selection ownership and de-highlight the widget. Activating the *select* widget twice is the only way to cause *xfontsel* to release the PRIMARY_FONT selection.

RESOURCES

The application class is **XFontSel**. Most of the user-interface is configured in the app-defaults file; if this file is missing a warning message will be printed to standard output and the resulting window will be nearly incomprehensible.

Most of the significant parts of the widget hierarchy are documented in the app-defaults file (normally /usr/lib/X11/app-defaults/XFontSel).

Application specific resources:

cursor (class **Cursor**)
: Specifies the cursor for the application window.

pattern (class **Pattern**)
: Specifies the font name pattern for selecting a subset of available fonts. Equivalent to the **-pattern** option. Most useful patterns will contain at least one field delimiter; e.g. "*-m-*" for monospaced fonts.

printOnQuit (class **PrintOnQuit**)
: If *True* the currently selected font name is printed to standard output when the quit button is activated. Equivalent to the **-print** option.

Widget specific resources:

showUnselectable (class **ShowUnselectable**)
: Specifies, for each field menu, whether or not to show values that are not currently selectable, based upon previous field selections. If shown, the unselectable values are clearly identified as such and do not highlight when the pointer is moved down the menu. The full name of this resource is **fieldN.menu.options.showUnselectable**, class **MenuButton.SimpleMenu.Options.ShowUnselectable**; where N is replaced with the field number (starting with the left-most field numbered 0). The default is True for all but field 11 (average width of characters in font) and False for field 11. If you never want to see unselectable entries, '*menu.options.showUnselectable:False' is a reasonable thing to specify in a resource file.

FILES

$XFILESEARCHPATH/XFontSel

SEE ALSO

xrdb(1)

BUGS

Sufficiently ambiguous patterns can be misinterpreted and lead to an initial selection string which may not correspond to what the user intended and which may cause the initial sample text output to fail to match the proffered string. Selecting any new field value will correct the sample output, though possibly resulting in no matching font.

Should be able to return a FONT for the PRIMARY selection, not just a STRING.

Any change in a field value will cause *xfontsel* to assert ownership of the PRIMARY_FONT selection. Perhaps this should be parameterized.

When running on a slow machine, it is possible for the user to request a field menu before the font names have been completely parsed. An error message indicating a missing menu is printed to stderr but otherwise nothing bad (or good) happens.

COPYRIGHT

AUTHOR

Ralph R. Swick, Digital Equipment Corporation/MIT Project Athena

NAME

xgc - X graphics demo

SYNOPSIS

xgc [-*toolkitoption* ...]

DESCRIPTION

The *xgc* program demonstrates various features of the X graphics primitives. Try the buttons, see what they do; we haven't the time to document them, perhaps you do?

OPTIONS

Xgc accepts all of the standard X Toolkit command line options.

X DEFAULTS

This program accepts the usual defaults for toolkit applications.

ENVIRONMENT

DISPLAY

to get the default host and display number.

XENVIRONMENT

to get the name of a resource file that overrides the global resources stored in the RESOURCE_MANAGER property.

SEE ALSO

X(1)

BUGS

This program isn't really finished yet.

COPYRIGHT

Copyright 1989, Massachusetts Institute of Technology.
See *X(1)* for a full statement of rights and permissions.

AUTHORS

Dan Schmidt, MIT

NAME

xhost - server access control program for X

SYNOPSIS

xhost [[+-]hostname ...]

DESCRIPTION

The *xhost* program is used to add and delete hosts to the list of machines that are allowed to make connections to the X server. This provides a rudimentary form of privacy control and security. It is only sufficient for a workstation (single user) environment, although it does limit the worst abuses. Environments which require more sophisticated measures should use the hooks in the protocol for passing authentication data to the server.

The server initially allows network connections only from programs running on the same machine or from machines listed in the file */etc/X*.hosts* (where * is the display number of the server). The *xhost* program is usually run either from a startup file or interactively to give access to other users.

Hostnames that are followed by two colons (::) are used in checking DECnet connections; all other hostnames are used for TCP/IP connections.

OPTIONS

Xhost accepts the following command line options described below. For security, the options that effect access control may only be run from the same machine as the server.

[+]*hostname*
: The given *hostname* (the plus sign is optional) is added to the list of machines that are allowed to connect to the X server.

–hostname
: The given *hostname* is removed from the list of machines that are allowed to connect to the server. Existing connections are not broken, but new connection attempts will be denied. Note that the current machine is allowed to be removed; however, further connections (including attempts to add it back) will not be permitted. Resetting the server (thereby breaking all connections) is the only way to allow local connections again.

+
: Access is granted to everyone, even if they aren't on the list of allowed hosts (i.e. access control is turned off).

–
: Access is restricted to only those machines on the list of allowed hosts (i.e. access control is turned on).

nothing
: If no command line arguments are given, the list of hosts that are allowed to connect is printed on the standard output along with a message indicating whether or not access control is currently enabled. This is the only option that may be used from machines

other than the one on which the server is running.

FILES

/etc/X*.hosts

SEE ALSO

X(1), Xserver(1)

ENVIRONMENT

DISPLAY

to get the default host and display to use.

BUGS

You can't specify a display on the command line because **–display** is a valid command line argument (indicating that you want to remove the machine named *''display''* from the access list).

COPYRIGHT

Copyright 1988, Massachusetts Institute of Technology.
See *X(1)* for a full statement of rights and permissions.

AUTHORS

Bob Scheifler, MIT Laboratory for Computer Science,
Jim Gettys, MIT Project Athena (DEC).

NAME

xinit - X Window System initializer

SYNOPSIS

xinit [[client] options] [-- [server] [display] options]

DESCRIPTION

The *xinit* program is used to start the X Window System server and a first client program (usually a terminal emulator) on systems that cannot start X directly from */etc/init* or in environments that use multiple window systems. When this first client exits, *xinit* will kill the X server and then terminate.

If no specific client program is given on the command line, *xinit* will look for a file in the user's home directory called *.xinitrc* to run as a shell script to start up client programs. If no such file exists, *xinit* will use the following as a default:

```
xterm -geometry +1+1 -n login -display :0
```

If no specific server program is given on the command line, *xinit* will look for a file in the user's home directory called *.xserverrc* to run as a shell script to start up the server. If no such file exists, *xinit* will use the following as a default:

```
X :0
```

Note that this assumes that there is a program named *X* in the current search path. However, servers are usually named *Xdisplaytype* where *displaytype* is the type of graphics display which is driven by this server. The site administrator should, therefore, make a link to the appropriate type of server on the machine, or create a shell script that runs *xinit* with the appropriate server.

An important point is that programs which are run by *.xinitrc* and by *.xserverrc* should be run in the background if they do not exit right away, so that they don't prevent other programs from starting up. However, the last long-lived program started (usually a window manager or terminal emulator) should be left in the foreground so that the script won't exit (which indicates that the user is done and that *xinit* should exit).

An alternate client and/or server may be specified on the command line. The desired client program and its arguments should be given as the first command line arguments to *xinit.* To specify a particular server command line, append a double dash (--) to the *xinit* command line (after any client and arguments) followed by the desired server comand.

Both the client program name and the server program name must begin with a slash (/) or a period (.). Otherwise, they are treated as an arguments to be appended to their respective startup lines. This makes it possible to add arguments (for example, foreground and background colors) without having to retype the whole command line.

If an explicit server name is not given and the first argument following the double dash (--) is a colon followed by a digit, *xinit* will use that number as the display number instead of zero. All remaining arguments are appended to the server command line.

EXAMPLES

Below are several examples of how command line arguments in *xinit* are used.

xinit This will start up a server named *X* and run the user's *.xinitrc*, if it exists, or else start an *xterm*.

xinit -- /usr/bin/X11/Xqdss :1
This is how one could start a specific type of server on an alternate display.

xinit -geometry =80x65+10+10 -fn 8x13 -j -fg white -bg navy
This will start up a server named *X*, and will append the given arguments to the default *xterm* command. It will ignore *.xinitrc*.

xinit -e widgets -- ./Xsun -l -c
This will use the command *./Xsun -l -c* to start the server and will append the arguments *-e widgets* to the default *xterm* command.

xinit /usr/ucb/rsh fasthost cpupig -display ws:1 -- :1 -a 2 -t 5
This will start a server named *X* on display 1 with the arguments *-a 2 -t 5*. It will then start a remote shell on the machine **fasthost** in which it will run the command *cpupig*, telling it to display back on the local workstation.

Below is a sample *.xinitrc* that starts a clock, several terminals, and leaves the window manager running as the ''last'' application. Assuming that the window manager has been configured properly, the user then chooses the ''Exit'' menu item to shut down X.

```
xrdb -load $HOME/.Xres
xsetroot -solid gray &
xclock -g 50x50-0+0 -bw 0 &
xload -g 50x50-50+0 -bw 0 &
xterm -g 80x24+0+0 &
xterm -g 80x24+0-0 &
uwm
```

Sites that want to create a common startup environment could simply create a default *.xinitrc* that references a site-wide startup file:

```
#!/bin/sh
. /usr/local/lib/site.xinitrc
```

Another approach is to write a script that starts *xinit* with a specific shell script. Such scripts are usually named *x11*, *xstart*, or *startx* and are a convenient way to provide a simple interface for novice users:

```
#!/bin/sh
xinit /usr/local/bin/startx -- /usr/bin/X11/Xhp :1
```

ENVIRONMENT VARIABLES

DISPLAY
This variable gets set to the name of the display to which clients should connect.

XINITRC
This variable specifies an init file containing shell commands to start up the initial windows. By default, *.xinitrc* in the home directory will be used.

SEE ALSO

X(1), Xserver(1), xterm(1), xrdb(1)

COPYRIGHT

Copyright 1988, Massachusetts Institute of Technology.
See *X(1)* for a full statement of rights and permissions.

AUTHOR

Bob Scheifler, MIT Laboratory for Computer Science

NAME

xkill - kill a client by its X resource

SYNOPSIS

xkill [–display *displayname*] [–id *resource*] [–button number] [–frame] [–all]

DESCRIPTION

Xkill is a utility for forcing the X server to close connections to clients. This program is very dangerous, but is useful for aborting programs that have displayed undesired windows on a user's screen. If no resource identifier is given with *-id*, *xkill* will display a special cursor as a prompt for the user to select a window to be killed. If a pointer button is pressed over a non-root window, the server will close its connection to the client that created the window.

OPTIONS

–display *displayname*
This option specifies the name of the X server to contact.

–id *resource*
This option specifies the X identifier for the resource whose creator is to be aborted. If no resource is specified, *xkill* will display a special cursor with which you should select a window to be kill.

–button *number*
This option specifies the number of pointer button that should be used in selecting a window to kill. If the word "any" is specified, any button on the pointer may be used. By default, the first button in the pointer map (which is usually the leftmost button) is used.

–all This option indicates that all clients with top-level windows on the screen should be killed. *Xkill* will ask you to select the root window with each of the currently defined buttons to give you several chances to abort. Use of this option is highly discouraged.

–frame This option indicates that xkill should ignore the standard conventions for finding top-level client windows (which are typically nested inside a window manager window), and simply believe that you want to kill direct children of the root.

XDEFAULTS

Button Specifies a specific pointer button number or the word "any" to use when selecting windows.

SEE ALSO

X(1), xwininfo(1), XKillClient and XGetPointerMapping in the Xlib Programmers Manual, KillClient in the X Protocol Specification

COPYRIGHT

AUTHOR

Jim Fulton, MIT X Consortium
Dana Chee, Bellcore

NAME

xload - load average display for X

SYNOPSIS

xload [*-toolkitoption* ...] [-scale *integer*] [-update *seconds*]

DESCRIPTION

The *xload* program displays a periodically updating histogram of the system load average. This program is nothing more than a wrapper around the Athena Load widget.

OPTIONS

Xload accepts all of the standard X Toolkit command line options along with the following additional options:

–scale *integer*

This option specifies the minimum number of tick marks in the histogram, where one division represents one load average point. If the load goes above this number, *xload* will create more divisions, but it will never use fewer than this number. The default is 1.

–update *seconds*

This option specifies the frequency in seconds at which *xload* updates its display. If the load average window is uncovered (by moving windows with a window manager or by the *xrefresh* program), the graph will be also be updated. The minimum amount of time allowed between updates is 5 seconds (which is also the default).

–hl *color*

This option specifies the color of the label and scale lines.

The following standard X Toolkit arguments are commonly used with *xload*:

–bd *color*

This option specifies the border color. The default is *black.*

–bg *color*

This option specifies the background color. The default is *white.*

–bw *pixels*

This option specifies the width in pixels of the border around the window. The default value is 2.

–fg *color*

This option specifies the graph color. The default color is *black.*

–fn *fontname*
: This option specifies the font to be used in displaying the name of the host whose load is being monitored. The default is *6x10.*

–rv
: This option indicates that reverse video should be simulated by swapping the foreground and background colors.

–geometry *geometry*
: This option specifies the prefered size and postion of the window.

–display *display*
: This option specifies the X server to contact.

–xrm *resourcestring*
: This option specifies a resource string to be used. This is especially useful for setting resources that do not have separate command line options.

X DEFAULTS

This program uses the *Load* widget in the X Toolkit. It understands all of the core resource names and classes as well as:

width (class **Width**)
: Specifies the width of the load average graph.

height (class **Height**)
: Specifies the height of the load average graph.

update (class **Interval**)
: Specifies the frequency in seconds at which the load should be redisplayed.

scale (class **Scale**)
: Specifies the initial number of ticks on the graph. The default is 1.

minScale (class **Scale**)
: Specifies the minimum number of ticks that will be displayed. The default is 1.

foreground (class **Foreground**)
: Specifies the color for the graph. The default is *black* since the core default for background is *white.*

highlight (class **Foreground**)
: Specifies the color for the text and scale lines. The default is the same as for the **foreground** resource.

label (class **Label**)
Specifies the label to use on the graph. The default is the hostname.

font (class **Font**)
Specifies the font to be used for the label. The default is *fixed.*

reverseVideo (class **ReverseVideo**)
Specifies that the foreground and background should be reversed.

SEE ALSO

X(1), xrdb(1), mem(4), Athena Load widget

DIAGNOSTICS

Unable to open display or create window. Unable to open /dev/kmem. Unable to query window for dimensions. Various X errors.

BUGS

This program requires the ability to open and read the special system file */dev/kmem.* Sites that do not allow general access to this file should make *xload* belong to the same group as */dev/kmem* and turn on the *set group id* permission flag.

Reading /dev/kmem is inherently non-portable. Therefore, the widget upon which this application is based must be ported to each new operating system.

Border color has to be explicitly specified when reverse video is used.

COPYRIGHT

Copyright 1988, Massachusetts Institute of Technology.
See *X(1)* for a full statement of rights and permissions.

AUTHORS

K. Shane Hartman (MIT-LCS) and Stuart A. Malone (MIT-LCS);
with features added by Jim Gettys (MIT-Athena), Bob Scheifler (MIT-LCS), and Tony Della Fera (MIT-Athena)

NAME

xlogo - X Window System logo

SYNOPSIS

xlogo [-*toolkitoption* ...]

DESCRIPTION

The *xlogo* program displays the X Window System logo. This program is nothing more than a wrapper around the Athena Logo widget.

OPTIONS

Xlogo accepts all of the standard X Toolkit command line options, of which the following are used most frequently:

–bg *color*
: This option specifies the color to use for the background of the window. The default is *white.* A correct color for the background is something like *maroon.*

–bd *color*
: This option specifies the color to use for the border of the window. The default is *black.*

–bw *number*
: This option specifies the width in pixels of the border surrounding the window.

–fg *color*
: This option specifies the color to use for displaying the logo. The default is *black.* A correct color for the background is something like *silver,* which you can approximate with a shade of gray, like #aa9.

–rv This option indicates that reverse video should be simulated by swapping the foreground and background colors.

–geometry *geometry*
: This option specifies the prefered size and position of the logo window.

–display *display*
: This option specifies the X server to contact.

–xrm *resourcestring*
: This option specifies a resource string to be used. This is especially useful for setting resources that do not have separate command line options.

X DEFAULTS

This program uses the *Logo* widget in the X Toolkit. It understands all of the core resource names and classes as well as:

width (class **Width**)
: Specifies the width of the logo. The default width is 100 pixels.

height (class **Height**)
: Specifies the height of the logo. The default height is 100 pixels.

foreground (class **Foreground)**
: Specifies the color for the logo. The default is *black* since the core default for background is *white.*

reverseVideo (class **ReverseVideo)**
: Specifies that the foreground and background should be reversed.

ENVIRONMENT

DISPLAY
: to get the default host and display number.

XENVIRONMENT
: to get the name of a resource file that overrides the global resources stored in the RESOURCE_MANAGER property.

SEE ALSO

X(1), xrdb(1), Athena Logo widget

COPYRIGHT

Copyright 1988, Massachusetts Institute of Technology.
See *X(1)* for a full statement of rights and permissions.

AUTHORS

Ollie Jones of Apollo Computer wrote the logo graphics routine, based on a graphic design by Danny Chong and Ross Chapman of Apollo Computer.

NAME

xlsatoms - list interned atoms defined on server

SYNOPSIS

xlsatoms [-options ...]

DESCRIPTION

Xlsatoms lists the interned atoms. By default, all atoms starting from 1 (the lowest atom value defined by the protocol) are listed until unknown atom is found. If an explicit range is given, *xlsatoms* will try all atoms in the range, regardless of whether or not any are undefined. used.

OPTIONS

–display *dpy*

This option specifies the X server to which to connect.

–format *string*

This option specifies a *printf*-style string used to list each atom *<value,name>* pair, printed in that order (*value* is an *unsigned long* and *name* is a *char* *). *Xlsatoms* will supply a newline at the end of each line. The default is *%ld\t%s*.

–range *[low]-[high]*

This option specifies the range of atom values to check. If *low* is not given, a value of 1 assumed. If *high* is not given, *xlsatoms* will stop at the first undefined atom at or above *low*.

–name *string*

This option specifies the name of an atom to list. If the atom does not exist, a message will be printed on the standard error.

SEE ALSO

X(1), Xserver(1), xprop(1)

ENVIRONMENT

DISPLAY

to get the default host and display to use.

COPYRIGHT

AUTHOR

Jim Fulton, MIT X Consortium

NAME

xlsclients - list client applications running on a display

SYNOPSIS

xlsclients [-display *displayname*] [-a] [-l] [-m maxcmdlen]

DESCRIPTION

Xlsclients is a utility for listing information about the client applications running on a display. It may be used to generate scripts representing a snapshot of the the user's current session.

OPTIONS

–display *displayname*
This option specifies the X server to contact.

–a This option indicates that clients on all screens should be listed. By default, only those clients on the default screen are listed.

–l This option indicates that a long listing showing the window name, icon name, and class hints in addition to the machine name and command string shown in the default listing.

–m *maxcmdlen*
This option specifies the maximum number of characters in a command to print out. The default is 1000.

ENVIRONMENT

DISPLAY
To get the default host, display number, and screen.

SEE ALSO

X(1), xwininfo(1), xprop(1)

COPYRIGHT

Copyright 1989, Massachusetts Institute of Technology.
See *X(1)* for a full statement of rights and permissions.

AUTHOR

Jim Fulton, MIT X Consortium

NAME

xlsfonts - server font list displayer for X

SYNOPSIS

xlsfonts [-options ...] [-fn pattern]

DESCRIPTION

Xlsfonts lists the fonts that match the given *pattern*. The wildcard character "*" may be used to match any sequence of characters (including none), and "?" to match any single character. If no pattern is given, "*" is assumed.

The "*" and "?" characters must be quoted to prevent them from being expanded by the shell.

OPTIONS

–display *host:dpy*
: This option specifies the X server to contact.

–l[l[l]]
: This option indicates that medium, long, and very long listings, respectively, should be generated for each font.

–m
: This option indicates that long listings should also print the minimum and maximum bounds of each font.

–C
: This option indicates that listings should use multiple columns. This is the same as **-n 0**.

–1
: This option indicates that listings should use a single column. This is the same as **-n 1**.

–w *width*
: This option specifies the width in characters that should be used in figuring out how many columns to print. The default is 79.

–n *columns*
: This option specifies the number of columns to use in displaying the output. By default, it will attempt to fit as many columns of font names into the number of character specified by **-w** *width.*

SEE ALSO

X(1), Xserver(1), xset(1), xfd(1)

ENVIRONMENT

DISPLAY
: to get the default host and display to use.

BUGS

Doing ‘‘xlsfonts -l’’ can tie up your server for a very long time. This is really a bug with single-threaded non-preemptible servers, not with this program.

COPYRIGHT

AUTHOR

Mark Lillibridge, MIT Project Athena; Jim Fulton, MIT X Consortium; Phil Karlton, SGI

NAME

xlswins - server window list displayer for X

SYNOPSIS

xlswins [-options ...] [windowid ...]

DESCRIPTION

Xlswins lists the window tree. By default, the root window is used as the starting point, although a specific window may be specified using the *-id* option. If no specific windows are given on the command line, the root window will be used.

OPTIONS

–display *displayname*
This option specifies the X server to contact.

–l
This option indicates that a long listing should be generated for each window. This includes a number indicating the depth, the geometry relative to the parent as well as the location relative to the root window.

–format *radix*
This option specifies the radix to use when printing out window ids. Allowable values are: *hex*, *octal*, and *decimal*. The default is hex.

–indent *number*
This option specifies the number of spaces that should be indented for each level in the window tree. The default is 2.

SEE ALSO

X(1), Xserver(1), xwininfo(1), xprop(1)

ENVIRONMENT

DISPLAY
to get the default host and display to use.

BUGS

This should be integrated with xwininfo somehow.

COPYRIGHT

AUTHOR

Jim Fulton, MIT X Consortium

NAME

xmag - magnify parts of the screen

SYNOPSIS

xmag [-option ...]

DESCRIPTION

The *xmag* program allows you to magnify portions of the screen. If no explicit region is specified, a square centered around the pointer is displayed indicating the area to be enlarged. Once a region has been selected, a window is popped up showing a blown up version of the region in which each pixel in the source image is represented by a small square of the same color. Pressing Button1 on the pointer in the enlargement window pops up a small window displaying the position, number, and RGB value of the pixel under the pointer until the button is released. Pressing the space bar or any other pointer button removes the enlarged image so that another region may be selected. Pressing "q", "Q", or "^C" in the enlargement window exits the program.

OPTIONS

–display *display*
: This option specifies the X server to use for both reading the screen and displaying the enlarged version of the image.

–geometry *geom*
: This option specifies the size and/or location of the enlargement window. By default, the size is computed from the size of the source region and the desired magnification. Therefore, only one of **–source** *size* and **–mag** *magfactor* options may be specified if a window size is given with this option.

–source *geom*
: This option specifies the size and/or location of the source region on the screen. By default, a 64x64 square centered about the pointer is provided for the user to select an area of the screen. The size of the source is used with the desired magnification to compute the default enlargement window size. Therefore, only one of **–geometry** *size* and **–mag** *magfactor* options may be specified if a source size is given with this option.

–mag *magfactor*
: This option specifies an integral factor by which the source region should be enlarged. The default magnification is 5. This is used with the size of the source to compute the default enlargement window size. Therefore, only one of **-geometry** *size* and **–source** *geom* options may be specified if a magnification factor is given with this option.

–bw *pixels*
: This option specifies the width in pixels of the border surrounding the enlargement window.

–bd *color*
: This option specifies the color to use for the border surrounding the enlargement window.

–bg *colororpixelvalue*
: This option specifies the name of the color to be used as the background of the enlargement window. If the name begins with a percent size (%), it is interpretted to be an absolute pixel value. This is useful when displaying large areas since pixels that are the same color as the background do not need to be painted in the enlargement. The default is to use the BlackPixel of the screen.

–fn *fontname*
: This option specifies the name of a font to use when displaying pixel values (used when Button1 is pressed in the enlargement window).

–z
: This option indicates that the server should be grabbed during the dynamics and the call to XGetImage. This is useful for ensuring that clients don't change their state as a result of entering or leaving them with the pointer.

X DEFAULTS

The *xmag* program uses the following X resources:

geometry (class **Geometry**)
: Specifies the size and/or location of the enlargement window.

source (class **Source**)
: Specifies the size and/or location of the source region on the screen.

magnification (class **Magnification**)
: Specifies the enlargement factor.

borderWidth (class **BorderWidth**)
: Specifies the border width in pixels.

borderColor (class **BorderColor**)
: Specifies the color of the border.

background (class **Background**)
: Specifies the color or pixel value to be used for the background of the enlargement window.

font (class **Font)**

Specifies the name of the font to use when displaying pixel values when the user presses Button1 in the enlargement window.

SEE ALSO

X(1), xwd(1)

BUGS

This program will behave strangely on displays that support windows of different depths.

Because the window size equals the source size times the magnification, you only need to specify two of the three parameters. This can be confusing.

Being able to drag the pointer around and see a dynamic display would be very nice.

Another possible interface would be for the user to drag out the desired area to be enlarged.

COPYRIGHT

Copyright 1988, Massachusetts Institute of Technology.

AUTHOR

Jim Fulton, MIT X Consortium

NAME

xman - Manual page display program for the X Window System.

SYNOPSIS

xman [-options ...]

DESCRIPTION

Xman is a manual page browser. The default size of the initial *xman* window is small so that you can leave it running throughout your entire login session. In the initial window there are three options: *Help* will pop up a window with on-line help, *Quit* will exit, and *Manual Page* will pop up a window with a manual page browser in it. You may pop up more than one manual page browser window from a single execution of *xman*.

For further information on using *xman* please read the on-line help information. The rest of this manual page will discuss customization of *xman*.

CUSTOMIZING XMAN

Xman allows customization of both the directories to be searched for manual pages, and the name that each directory will map to in the *Sections* menu. Xman determines which directories it will search by reading the *MANPATH* environment variable. If no *MANPATH* is found then the directory is /usr/man is searched on POSIX systems. This environment is expected to be a colon-separated list of directories for xman to search.

```
setenv MANPATH /mit/kit/man:/usr/man
```

By default, *xman* will search each of the following directories (in each of the directories specified in the users MANPATH) for manual pages. If manual pages exist in that directory then they are added to list of manual pages for the corresponding menu item. A menu item is only displayed for those sections that actually contain manual pages.

Directory	Section Name
---------	------------
man1	(1) User Commands
man2	(2) System Calls
man3	(3) Subroutines
man4	(4) Devices
man5	(5) File Formats
man6	(6) Games
man7	(7) Miscellaneous
man8	(8) Sys. Administration
manl	(l) Local
mann	(n) New
mano	(o) Old

For instance, a user has three directories in her manual path and each contain a directory called *man3*. All these manual pages will appear alphabetically sorted when the user selects the menu item called *(3) Subroutines*. If there is no directory called *mano* in any of the directories in her MANPATH, or there are no manual pages in any of the directories called *mano* then no menu item will be displayed for the section called *(o) Old*.

By using the *mandesc* file a user or system manager is able to more closely control which manual pages will appear in each of the sections represented by menu items in the *Sections* menu. This functionality is only available on a section by section basis, and individual manual pages may not be handled in this manner (Although generous use of symbolic links - ln(1) - will allow almost any configuration you can imagine).

The format of the mandesc file is a character followed by a label. The character determines which of the sections will be added under this label. For instance suppose that you would like to create an extra menu item that contains all programmer subroutines. This label should contain all manual pages in both sections two and three. The *mandesc* file would look like this:

```
2Programmer Subroutines
3Programmer Subroutines
```

This will add a menu item to the *Sections* menu that would bring up a listing of all manual pages in sections two and three of the Programmers Manual. Since the label names are *exactly* the same they will be added to the same section. Note, however, that the original sections still exist.

If you want to completely ignore the default sections in a manual directory then add the line:

```
no default sections
```

anywhere in your mandesc file. This keeps xman from searching the default manual sections *In that directory only*. As an example, suppose you want to do the same thing as above, but you don't think that it is useful to have the *System Calls* or *Subroutines* sections any longer. You would need to duplicate the default entries, as well as adding your new one.

```
no default sections
1(1) User Commands
2Programmer Subroutines
3Programmer Subroutines
4(4) Devices
```

5(5) File Formats
6(6) Games
7(7) Miscellaneous
8(8) Sys. Administration
l(l) Local
n(n) New
o(o) Old

Xman will read any section that is of the from *man<character>*, where <character> is an upper or lower case letter (they are treated distinctly) or a numeral (0-9). Be warned, however, that man(1) and catman(8) will not search directories that are non-standard.

COMMAND LINE OPTIONS

Xman supports all standard Toolkit command line arguments (see *X(1)*). The following additional arguments are supported.

-helpfile *filename*
: Specifies a helpfile to use other than the default.

-bothshown
: Allows both the manual page and manual directory to be on the screen at the same time.

-notopbox
: Starts without the Top Menu with the three buttons in it.

-geometry *WxH+X+Y*
: Sets the size and location of the Top Menu with the three buttons in it.

-pagesize *WxH+X+Y*
: Sets the size and location of all the Manual Pages.

WIDGETS

In order to specify resources, it is useful to know the hierarchy of the widgets which compose *xman*. In the notation below, indentation indicates hierarchical structure. The widget class name is given first, followed by the widget instance name.

```
Xman xman        (This widget is never used)
          TopLevelShell  topbox
                    Form  form
                              Label  topLabel
                              Command  helpButton
                              Command  quitButton
                              Command  manpageButton
```

```
                        TransientShell search
                                DialogWidgetClass dialog
                                        Label label
                                        Text value
                                        Command manualPage
                                        Command apropos
                                        Command cancel
                        TransientShell pleaseStandBy
                                Label label
                TopLevelShell manualBrowser
                        Paned Manpage_Vpane
                                Paned horizPane
                                        MenuButton options
                                        MenuButton sections
                                        Label manualBrowser
                                Viewport directory
                                        List directory
                                        List directory
                                        .
                                        . (one for each section,
                                        . created "on the fly")
                                        .
                                ScrollByLine manualPage
                        SimpleMenu optionMenu
                                SmeBSB displayDirectory
                                SmeBSB displayManualPage
                                SmeBSB help
                                SmeBSB search
                                SmeBSB showBothScreens
                                SmeBSB removeThisManpage
                                SmeBSB openNewManpage
                                SmeBSB showVersion
                                SmeBSB quit
                        SimpleMenu sectionMenu
                                SmeBSB <name of section>
                                        .
                                        . (one for each section)
                                        .
                        TransientShell search
                                DialogWidgetClass dialog
                                        Label label
                                        Text value
                                        Command manualPage
                        Command apropos
                        Command cancel
        TransientShell pleaseStandBy
                Label label
        TransientShell likeToSave
                Dialog dialog
                        Label label
                        Text value
                        Command yes
                        Command no
TopLevelShell help
        Paned Manpage_Vpane
                Paned horizPane
                        MenuButton options
                        MenuButton sections
                        Label manualBrowser
                ScrollByLine manualPage
        SimpleMenu optionMenu
                SmeBSB displayDirectory
                SmeBSB displayManualPage
                SmeBSB help
                SmeBSB search
                SmeBSB showBothScreens
                SmeBSB removeThisManpage
                SmeBSB openNewManpage
                SmeBSB showVersion
                SmeBSB quit
```

APPLICATION RESOURCES

xman has the following application-specific resources which allow customizations unique to *xman*.

manualFontNormal (Class **Font**)
The font to use for normal text in the manual pages.

manualFontBold (Class **Font**)
The font to use for bold text in the manual pages.

manualFontItalic (Class **Font**)
The font to use for italic text in the manual pages.

directoryFontNormal (Class **Font**)
The font to use for the directory text.

bothShown (Class **Boolean**)
Either 'true' or 'false', specifies whether or not you want both the directory and the manual page shown at start up.

directoryHeight (Class **DirectoryHeight**)
The height in pixels of the directory, when the directory and the manual page are shown simultaneously.

topCursor (Class **Cursor**)
The cursor to use in the top box.

helpCursor (Class **Cursor**)
The cursor to use in the help window.

manpageCursor (Class **Cursor**)
The cursor to use in the manual page window.

searchEntryCursor (Class **Cursor**)
The cursor to use in the search entry text widget.

pointerColor (Class **Foreground**)
This is the color of all the cursors (pointers) specified above. The name was chosen to be compatible with xterm.

helpFile (Class **File**) Use this rather than the system default helpfile.

topBox (Class **Boolean**)
Either 'true' or 'false', determines whether the top box (containing the help, quit and manual page buttons) or a manual page is put on the screen at start-up. The default is true.

verticalList (Class **Boolean**)
Either 'true' or 'false', determines whether the directory listing is vertically or horizontally organized. The default is horizontal (false).

GLOBAL ACTIONS

Xman defines all user interaction through global actions. This allows the user to modify the translation table of any widget, and bind any event to the new user action. The list of actions supported by *xman* are:

GotoPage(*page*) When used in a manual page display window this will allow the user to move between a directory and manual page display. The *page* argument can be either **Directory** or **ManualPage**.

Quit() This action may be used anywhere, and will exit xman.

Search(*type*, *action*) Only useful when used in a search popup, this action will cause the search widget to perform the named search type on the string in the search popup's value widget. This action will also pop down the search widget. The *type* argument can be either **Apropos, Manpage** or **Cancel.** If an *action* of **Open** is specified then xman will open a new manual page to display the results of the search, otherwise xman will attempt to display the results in the parent of the search popup.

PopupHelp() This action may be used anywhere, and will popup the help widget.

PopupSearch() This action may be used anywhere except in a help window. It will cause the search popup to become active and visible on the screen, allowing the user search for a manual page.

CreateNewManpage() This action may be used anywhere, and will create a new manual page display window.

RemoveThisManpage() This action may be used in any manual page or help display window. When called it will remove the window, and clean up all resources associated with it.

SaveFormattedPage(*action*)
This action can only be used in the "likeToSave" popup widget, and tells xman whether to **Save** or **Cancel** a save of the manual page that has just been formatted.

ShowVersion() This action may be called from any manual page or help display window, and will cause the informational display line to show the current version of xman.

FILES

<*manpath directory*>/man<*character*>

<*manpath directory*>/cat<*character*>

<*manpath directory*>/mandesc

/usr/lib/X11/app-defaults/Xman	specifies required resources
/tmp	*Xman* creates temporary files in /tmp for all unformatted man pages and all apropos searches.

SEE ALSO

X(1), X(8C), man(1), apropos(1), catman(8), Athena Widget Set

ENVIRONMENT

DISPLAY	the default host and display to use.
MANPATH	the search path for manual pages. Directories are separated by colons (e.g. /usr/man:/mit/kit/man:/foo/bar/man).
XENVIRONMENT	to get the name of a resource file that overrides the global resources stored in the RESOURCE_MANAGER property.
XAPPLRESDIR	A string that will have "Xman" appended to it. This string will be the full path name of a user app-defaults file to be merged into the resource database after the system app-defaults file, and before the resources that are attached to the display.

BUGS

There probably are some.

COPYRIGHT

AUTHORS

Chris Peterson, MIT X Consortium from the V10 version written by Barry Shein formerly of Boston University.

NAME

xmessage - X window system message display program.

SYNOPSIS

xmessage [-options ...] -m <message>

DESCRIPTION

Xmessage opens a window on the screen that will contain the text of a message from either the command line or stdin. This text may have a scroll bar along the left side to allow the user to browse through relatively long messages. Along the lower edge of the message is list of words with boxes around them, clicking the left mouse button on any of these "buttons" (words with boxes around them) will cause the message to go away. If there is more than one "button" then some state will be returned to the invoker of the xmessage process via a change of the exit status of the program.

This program serves two functions, firstly it can be a method for shell scripts to present the user with information much as 'echo' allows in a tty environment, as well as allowing the user you answer simple questinos. Secondly it allows much of the functionality of 'cat' again in a windowing verses tty environment.

It should be noted that this program is intended for short messages, and will be quite slow when asked to handle long files from stdin. Although xmessage can accept input from stdin, this input is not allowed to come from a tty, if this is attempted, and error message will be printed. If xmessage is executed with an incorrect argument then it will print a usage message to standard out, as well as to an xmessage window.

COMMAND LINE OPTIONS

These are the command line options that xmessage understands. Please note that some of these are inherited from the XToolkit and as the list of default toolkit options changes xman will follow.

-printlabel

This will case the program to print the label of the button pressed to standard output (stdout), I envision this to be useful when popping up a message to a friend, as in: "ready to go to lunch". This allows you to know if he clicked the yes or the no button.

-noscroll (-nsb)

The scroll bar is active on the text window by default, this causes it to be removed.

-buttons <button> <button> ...
This option will cause xmessage to create one button for each arguement the follows it until something starts with a '-'. The string passed to the button is the name of the Command button widget created and will be the default text displayed to the user. Since this is the name of the widget it may be used to change any of the Xresources associated with that button.

-message <word> <word> ...
This must be the last argument in the command list, as every argument after this one is assumed to be part of the message. There is no limit to the length of this message.

-geometery (height)x(width)+(x_offset)+(y_offset)
Sets the size and location of the window created by xmessage.

-bw <pixels>

-borderwidth <pixels>
Specifies the width of the border for all windows in xmessage.

-bd <color>

-bordercolor <color>
Specifies the color of the borders of all windows in xmessage.

-fg <color>

-foreground <color>
Specifies the foreground color to be used.

-bg <color>

-background <color>
Specifies the background color to be used.

-fn <font>

-font <font>
Specifies the font to use for all buttons and text.

-display <host:display[.screen]>
Specifies a display other than the default specified by the DISPLAY environment variable to use to use.

-name <name>
Specifies the name to use when retrieving resources.

-title <title>
Specifies the title of this application.

-xrm <resource>
Allows a resource to be specified on the command line.

WIDGET AND RESOURCE NAMES

Resource management is an important part of X Toolkit applications, and xmessage is not exception, all objects in xmessage can have many of their distinguishing characteristics changed by changing the resources associated with them, below is a brief list of the resources and what they modify.

foreground	- foreground color
background	- background color
width & height	- size
borderWidth	- border width
borderColor	- border color

In order to change the default values for the widget resources you need to have the names, thus, below I have specified the names of some of the most common widgets.

xmessage - (argv[0])	- shell widget that contains everything displayed.
text	- the text window.
<button name>	- each of the buttons. "okay" is default.

You can also reference Widgets by class, The important classes for this application are: Command and Text.

Here are a few example of how to string all this information together into a resource specification, that can be used on the command line with the -xrm flag, or added to your .Xresource file.

xmessage*Command.foreground: Blue	All command buttons will be blue.
xmessage*foreground: Blue	Everything in the xmessage window has a blue foreground.
xmessage*Text.border: Red	The text widget has a red border.

In addition Xmessage has a few specific application resources, that allow customizations that are specific to xmessage.

ScrollText
A Boolean reasource that determines whether you are allowed to scroll the text widget the default value is TRUE.

printLabel
A Boolean resource that determines whether or not the label of the buton pressed to exit the program is printed, default value is FALSE.

ERROR MESSAGES

Xmessage errors may be printed into their owm xmessage window, this invocation of xmessage has a different name. This allows its resources to be specified seperatly, the name of xmessage error program is xmessage_error.

EXIT STATUS

Xmessage will exit with status zero (0) when there is only one button in the list, and it is clicked to exit. If there is more than one button in the list then the exit status will corrospond the number of the button pressed, starting at one hundred and one (101) for the first button, and counting up. Zero (0) is not used because no button should have a prefered place over the others.

SEE ALSO

X(1), X(8C), echo(1), cat(1)

BUGS

There must be some, somewhere.

AUTHORS

Copyright 1988 by Massachusetts Institute of Technology.
Chris Peterson, MIT Project Athena

NAME

xmh – X interface to the MH message handling system

SYNOPSIS

xmh [-path *mailpath*] [-initial *foldername*] [-flag] [-*toolkitoption* ...]

DESCRIPTION

The *xmh* program provides a window-oriented user interface to the Rand *MH* Message Handling System. To actually do things with your mail, it makes calls to the *MH* package. Electronic mail messages may be composed, sent, received, replied to, forwarded, sorted, and stored in folders.

To specify an alternate collection of mail folders in which to process mail, use **-path** followed by the pathname of the alternate mail directory. The default mail path is the value of the Path component in $HOME/.mh_profile, or $HOME/Mail if the *MH* Path is not given. To specify an alternate folder which may receive new mail and is initially opened by *xmh*, use the **-initial** flag. The default initial folder is 'inbox'. The option **-flag** will cause *xmh* to attempt to change the appearance of its icon when new mail has arrived. These three options have corresponding application-specific resources, named **MailPath**, **InitialFolder**, and **MailWaitingFlag**, which can be used in a resource file. The standard toolkit command line options are given in *X(1)*.

Please don't be misled by the size of this document. It introduces many aspects of the Athena Widget Set, and provides extensive mechanism for customization of the user interface. *xmh* really is easy to use.

INSTALLATION

The current version of *xmh* requires that the user is already set up to use *MH*, version 6. To do so, see if there is a file called .mh_profile in your home directory. If it exists, check to see if it contains a line that starts with ''Current-Folder''. If it does, you've been using version 4 or earlier of *MH*; to convert to version 6, you must remove that line. (Failure to do so causes spurious output to stderr, which can hang *xmh* depending on your setup.)

If you do not already have a .mh_profile, you can create one (and everything else you need) by typing ''inc'' to the shell. You should do this before using *xmh* to incorporate new mail.

For more information, refer to the *mh(1)* documentation.

BASIC SCREEN LAYOUT

xmh starts out with a single window, divided into four main areas:

- Six buttons with pull-down command menus.
- A collection of buttons, one for each top level folder. New users of mh will have two folders, ''drafts'' and ''inbox''.
- A listing, or Table of Contents, of the messages in the open folder. Initially, this will show the messages in ''inbox''.
- A view of one of your messages. Initially this is blank.

XMH AND THE ATHENA WIDGET SET

xmh uses the X Toolkit Intrinsics and the Athena Widget Set. Many of the features described below (scrollbars, buttonboxes, etc.) are actually part of the Athena Widget Set, and are described here only for completeness. For more information, see the Athena Widget Set documentation.

SCROLLBARS

Some parts of the main window will have a vertical area on the left containing a grey bar. This area is a *scrollbar*. They are used whenever the data in a window takes up more space than can be displayed. The grey bar indicates what portion of your data is visible. Thus, if the entire length of the area is grey, then you are looking at all your data. If only the first half is grey, then you are looking at the top half of your data. The message viewing area will have a horizontal scrollbar if the text of the message is wider than the viewing area.

You can use the pointer in the scrollbar to change what part of the data is visible. If you click with the middle button, then the top of the grey area will move to where the pointer is, and the corresponding portion of data will be displayed. If you hold down the middle button, you can drag around the grey area. This makes it easy to get to the top of the data: just press with the middle, drag off the top of the scrollbar, and release.

If you click with button 1, then the data to the right of the pointer will scroll to the top of the window. If you click with pointer button 3, then the data at the top of the window will scroll down to where the pointer is.

BUTTONBOXES, BUTTONS, AND MENUS

Any area containing many words or short phrases, each enclosed in a rectangle or rounded boundary, is called a *buttonbox*. Each rectangle or rounded area is actually a button that you can press by moving the pointer onto it and pressing pointer button 1. If a given buttonbox has more buttons in it than can fit, it will be displayed with a scrollbar, so you can always scroll to the button you want.

Some buttons have pull-down menus. Pressing the pointer button while the pointer is over one of these buttons will pull down a menu. Holding the button down while moving the pointer over the menu, called dragging the pointer, will highlight each selectable item on the menu as the pointer passes over it. To select an item in the menu, release the pointer button while the item is highlighted.

ADJUSTING THE RELATIVE SIZES OF AREAS

If you're not satisfied with the sizes of the various areas of the main window, they can easily be changed. Near the right edge of the border between each region is a black box, called a *grip*. Simply point to that grip with the pointer, press a pointer button, drag up or down, and release. Exactly what happens depends on which pointer button you press.

If you drag with the middle button, then only that border will move. This mode is simplest to understand, but is the least useful.

If you drag with pointer button 1, then you are adjusting the size of the window above. *xmh* will attempt to compensate by adjusting some window below it.

If you drag with pointer button 3, then you are adjusting the size of the window below. *xmh* will attempt to compensate by adjusting some window above it.

All windows have a minimum and maximum size; you will never be allowed to move a border past the point where it would make a window have an invalid size.

PROCESSING YOUR MAIL

This section will define the concepts of the selected folder, current folder, selected message(s), current message, selected sequence, and current sequence. Each *xmh* command is introduced.

For use in customization, action procedures corresponding to each command are given; these action procedures can be used to customize the user interface, particularly the keyboard accelerators and the functionality of the buttons in the optional button box created by the application resource **CommandButtonCount.**

SELECTED FOLDER

A folder contains a collection of mail messages, or is empty.

The selected folder is whichever foldername appears in the bar above the folder buttons. Note that this is not necessarily the same folder that is being viewed. To change the selected folder, just press on the desired folder button; if that folder has subfolders, select a folder from the pull down menu.

The Table of Contents, or toc, lists the messages in the viewed folder. The title bar above the Table of Contents displays the name of the viewed folder.

The toc title bar also displays the name of the viewed sequence of messages within the viewed folder. Every folder has an ``all'' sequence, which contains all the messages in the folder, and initially the toc title bar will show ``inbox:all''.

FOLDER COMMANDS

The *folder* command menu contains commands of a global nature:

Open Folder

Display the data in the selected folder. Thus, the selected folder also becomes the viewed folder. The action procedure corresponding to this command is **XmhOpenFolder**([*foldername*]). It takes an optional argument as the name of a folder to select and open; if no folder is specified, the selected folder is opened. It may be specified as part of an event translation from a folder menu button or from a folder menu, or as a binding of a keyboard accelerator to any widget other than the folder menu buttons or the folder menus.

Open Folder in New Window

Displays the selected folder in an additional main window. Note, however, that you may not reliably display the same folder in more than one window at a time, although *xmh* will not prevent you from trying. The corresponding action is **XmhOpenFolderInNewWindow()**.

Create Folder

Create a new folder. You will be prompted for a name for the new folder; to enter the name, move the pointer to the blank box provided and type. Subfolders are created by specifying the parent folder, a slash, and the subfolder name. For example, to create a folder named ``xmh'' which is a subfolder of an existing folder named ‘‘clients’’, type ‘‘clients/xmh’’. Click on the Okay button when finished, or just type Return; click on Cancel to cancel this operation. The action corresponding to Create Folder is **XmhCreateFolder()**.

Delete Folder

Destroy the selected folder. You will be asked to confirm this action (see CONFIRMATION WINDOWS). Destroying a folder will also destroy any subfolders of that folder. The corresponding action is **XmhDeleteFolder()**.

Close Window

Exits *xmh*, after first confirming that you won't lose any changes; or, if selected from any additional *xmh* window, simply closes that window. The corresponding action is **XmhClose()**.

HIGHLIGHTED MESSAGES, SELECTED MESSAGES AND THE CURRENT MESSAGE

It is possible to highlight a set of adjacent messages in the area of the Table of Contents. To highlight a message, click on it with pointer button 1. To highlight a range of messages, click on the first one with pointer button 1 and on the last one with pointer button 3; or press pointer button 1, drag, and release. To extend a range of selected messages, use pointer button 3. To highlight all messages in the table of contents, click rapidly three times with pointer button 1. To cancel any selection in the table of contents, click rapidly twice.

The selected messages are the same as the highlighted messages, if any. If no messages are highlighted, then the selected messages are considered the same as the current message.

The current message is indicated by a '+' next to the message number. It usually corresponds to the message currently being viewed. When a message is viewed, the title bar above the view will identify the message.

TABLE OF CONTENTS COMMANDS

The *Table of Contents* command menu contains commands which operate on the open, or viewed folder.

Incorporate New Mail

Add any new mail received to your inbox folder, and set the current message to be the first new message. (This command is selectable only if ‘‘inbox’’ is the folder being viewed.) The corresponding action is

XmhIncorporateNewMail().

Commit Changes Execute all deletions, moves, and copies that have been marked in this folder. The corresponding action is **XmhCommitChanges()**.

Pack Folder Renumber the messages in this folder so they start with 1 and increment by 1. The corresponding action is **XmhPackFolder()**.

Sort Folder Sort the messages in this folder in chronological order. As a side effect, this also packs the folder. The corresponding action is **XmhSortFolder()**.

Rescan Folder Rebuild the list of messages. This can be used whenever you suspect that *xmh*'s idea of what messages you have is wrong. (In particular, this is necessary if you change things using straight *MH* commands without using *xmh*.) The corresponding action is **XmhForceRescan()**.

MESSAGE COMMANDS

The *Message* command menu contains commands which operate on the selected message(s), or if there are no selected messages, the current message.

Compose Message Composes a new message. A new window will be brought up for composition; a description of it is given in the COMPOSITION WINDOWS section below. This command does not affect the current message. The corresponding action is **XmhComposeMessage()**.

View Next Message View the first selected message. If no messages are highlighted, view the current message. If current message is already being viewed, view the first unmarked message after the current message. The corresponding action is **XmhViewNextMessage()**.

View Previous View the last selected message. If no messages are highlighted, view the current message. If current message is already being viewed, view the first unmarked message before the current message. The corresponding action is **XmhViewPrevious()**.

Mark Deleted Mark the selected messages for deletion. If no messages are highlighted, then this mark the current message for deletion and automatically display the next unmarked message. The corresponding action is **XmhMarkDeleted()**.

Mark Move Mark the selected messages to be moved into the current (selected) folder. (If the current folder is the same as the viewed folder, this command will just beep.) If no messages are highlighted, this will mark the current message to be moved and display the next unmarked message. The corresponding action is **XmhMarkMove()**.

Mark Copy Mark the selected messages to be copied into the current folder. (If the current folder is the same as the viewed folder, this command will just beep.) If no messages are highlighted, mark the current message to be copied. The corresponding action is **XmhMarkCopy()**.

Unmark Remove any of the above three marks from the selected messages, or the current message, if none are highlighted. The corresponding action is **XmhUnmark()**.

View in New Window
Create a new window containing only a view of the first selected message, or the current message, if none are highlighted. The corresponding action is **XmhViewInNewWindow()**.

Reply Create a composition window in reply to the first selected message, or the current message, if none are highlighted. The corresponding action is **XmhReply()**.

Forward Create a composition window whose body is initialized to be the contents of the selected messages, or the current message if none are highlighted. The corresponding action is **XmhForward()**.

Use as Composition Create a composition window whose body is initialized to be the contents of the first selected message, or the current message if none are selected. Any changes you make in the composition will be saved in a new message in the ''drafts'' folder, and will not change the original message. However, this command was designed to be used within the ''drafts'' folder to compose message drafts, and there is an exception to this rule. If the message to be used as composition was selected from the ''drafts'' folder, the changes will be reflected in the original message (see COMPOSITION WINDOWS). The action procedure corresponding to this command is **XmhUseAsComposition()**.

Print Print the selected messages, or the current message if none are selected. *xmh* normally prints by invoking the *enscript*(1) command, but this can be customized with the application-specific resource **PrintCommand**. The action procedure corresponding to this command is **XmhPrint()**.

SEQUENCE COMMANDS

The *Sequence* command menu contains commands pertaining to message sequences (See MESSAGE-SEQUENCES), and a list of the message-sequences defined for the currently viewed folder. The selected message-sequence is indicated by a check mark in its entry in the margin of the menu. To change the selected message-sequence, select a new message-sequence from the sequence menu.

Pick Messages Define a new message-sequence. The corresponding action is **XmhPickMessages()**.

The following menu entries will be sensitive only if the current folder has any message-sequences other than the ''all'' message-sequence.

Open Sequence Change the viewed sequence to be the same as the selected sequence. The corresponding action is **XmhOpenSequence()**.

Add to Sequence Add the selected messages to the selected sequence. The corresponding action is **XmhAddToSequence()**.

Remove from Sequence
Remove the selected messages from the selected sequence. The corresponding action is **XmhRemoveFromSequence()**.

Delete Sequence Remove the selected sequence entirely. The messages themselves are not affected; they simply are no longer grouped together to define a message-sequence. The corresponding action is

XmhDeleteSequence().

VIEW COMMANDS

Commands in the View menu and in the buttonboxes of view windows (which result from the Message command "View In New") correspond in functionality to commands of the same name in the Message menu, but they operate on the viewed message rather than the selected messages or current message.

Close Window	When the viewed message is in a separate view window, this command will close the view, after confirming the status of any unsaved edits. The corresponding action procedure is **XmhCloseView()**.
Reply	Create a composition window in reply to the viewed message. The related action procedure is **XmhViewReply()**.
Forward	Create a composition window whose body is initialized to be the contents of the viewed message. The corresponding action is **XmhViewForward()**.
Use As Composition	Create a composition window whose body is initialized to be the contents of the viewed message. Any changes made in the composition window will be saved in a new message in the "drafts" folder, and will not change the original message. An exception: if the viewed message was selected from the "drafts" folder, the original message is edited. The action procedure corresponding to this command is **XmhViewUseAsComposition()**.
Edit Message	This command enables the direct editing of the viewed message. The action procedure is **XmhEditView()**.
Save Message	This command is insensitive until the message has been edited; when activated, edits will be saved to the original message in the view. The corresponding action is **XmhSaveView()**.
Print	Print the viewed message. *xmh* prints by invoking the *enscript*(1) command, but this can be customized with the application-specific resource **PrintCommand**. The corresponding action procedure is **XmhPrintView()**.

OPTIONS

The *Options* menu contains one entry.

Read in Reverse

When selected, a check mark appears in the margin of this menu entry. Read in Reverse will switch the meaning of the next and previous messages, and will increment in the opposite direction. This is useful if you want to read your messages in the order of most recent first. The option acts as a toggle; select it from the menu a second time to undo the effect. The check mark appears when the option is selected.

COMPOSITION WINDOWS

Aside from the normal text editing functions, there are six command buttons associated with composition windows:

Close Window	Close this composition window. If changes have been made since the most recent Save or Send, you will be asked to confirm losing them. The corresponding action is **XmhCloseView()**.
Send	Send this composition. The corresponding action is **XmhSend()**.
New Headers	Replace the current composition with an empty message. If changes have been made since the most recent Send or Save, you will be asked to confirm losing them. The corresponding action is **XmhResetCompose()**.
Compose Message	Bring up another new composition window. The corresponding action is **XmhComposeMessage()**.
Save Message	Save this composition in your drafts folder. Then you can safely close the composition. At some future date, you can continue working on the composition by opening the drafts folder, selecting the message, and using the ``Use as Composition'' command. The corresponding action is **XmhSave()**.
Insert	Insert a related message into the composition. If the composition window was created with a ``Reply'' command, the related message is the message being replied to, otherwise no related message is defined and this button is insensitive. The message may be filtered before being inserted; see **ReplyInsertFilter**

under APPLICATION RESOURCES for more information. The corresponding action is **XmhInsert()**.

ACCELERATORS

Accelerators are shortcuts. They allow you to invoke commands without using the menus, either from the keyboard or by using the pointer.

xmh defines pointer accelerators for common actions: To select and view a message with a single click, use pointer button 2 on the message's entry in the table of contents. To select and open a folder or a sequence in a single action, make the folder or sequence selection with pointer button 2.

To mark the highlighted messages to be moved in a single action, or current message if none have been highlighted, use pointer button 3 to select the target folder. Similarly, selecting a sequence with pointer button 3 will add the highlighted or current message(s) to that sequence. In both of these operations, the selected folder or sequence and the viewed folder or sequence are not changed.

xmh defines the following keyboard accelerators over the surface of the main window, except in the view area while editing a message:

Meta-I	Incorporate New Mail
Meta-C	Commit Changes
Meta-R	Rescan Folder
Meta-P	Pack Folder
Meta-S	Sort Folder
Meta-space	View Next Message
Meta-c	Mark Copy
Meta-d	Mark Deleted
Meta-f	Forward the selected or current message
Meta-m	Mark Move
Meta-n	View Next Message
Meta-p	View Previous Message
Meta-r	Reply to the selected or current message
Meta-u	Unmark
Ctrl-V	Scroll the table of contents forward
Meta-V	Scroll the table of contents backward
Ctrl-v	Scroll the view forward
Meta-v	Scroll the view backward

TEXT EDITING COMMANDS

All of the text editing commands are actually defined by the Text widget in the Athena Widget Set. The commands may be bound to different keys than the defaults described below through the X Toolkit Intrinsics key re-binding mechanisms. See the X Toolkit Intrinsics and the Athena Widget Set documentation for more details.

Whenever you are asked to enter any text, you will be using a standard text editing interface. Various control and meta keystroke combinations are bound to a somewhat Emacs-like set of commands. In addition, the pointer buttons may be used to select a portion of text or to move the insertion point in the text. Pressing pointer button 1 causes the insertion point to move to the pointer. Double-clicking button 1 selects a word, triple-clicking selects a line, quadruple-clicking selects a paragraph, and clicking rapidly five times selects everything. Any selection may be extended in either direction by using pointer button 3.

In the following, a *line* refers to one displayed row of characters in the window. A *paragraph* refers to the text between carriage returns. Text within a paragraph is broken into lines for display based on the current width of the window. When a message is sent, text is broken into lines based upon the values of the **SendBreakWidth** and **SendWidth** application-specific resources.

The following keystroke combinations are defined:

Ctrl-a	Beginning Of Line	Meta-b	Backward Word
Ctrl-b	Backward Character	Meta-f	Forward Word
Ctrl-d	Delete Next Character	Meta-i	Insert File
Ctrl-e	End Of Line	Meta-k	Kill To End Of Paragraph
Ctrl-f	Forward Character	Meta-q	Form Paragraph
Ctrl-g	Multiply Reset	Meta-v	Previous Page
Ctrl-h	Delete Previous Character	Meta-y	Insert Current Selection
Ctrl-j	Newline And Indent	Meta-z	Scroll One Line Down
Ctrl-k	Kill To End Of Line	Meta-d	Delete Next Word
Ctrl-l	Redraw Display	Meta-D	Kill Word
Ctrl-m	Newline	Meta-h	Delete Previous Word
Ctrl-n	Next Line	Meta-H	Backward Kill Word
Ctrl-o	Newline And Backup	Meta-<	Beginning Of File
Ctrl-p	Previous Line	Meta->	End Of File
Ctrl-r	Search/Replace Backward	Meta-]	Forward Paragraph
Ctrl-s	Search/Replace Forward	Meta-[	Backward Paragraph
Ctrl-t	Transpose Characters		
Ctrl-u	Multiply by 4	Meta-Delete	Delete Previous Word
Ctrl-v	Next Page	Meta-Shift Delete	Kill Previous Word

Ctrl-w	Kill Selection	Meta-Backspace	Delete Previous Word
Ctrl-y	Unkill	Meta-Shift Backspace	Kill Previous Word
Ctrl-z	Scroll One Line Up		

In addition, the pointer may be used to cut and paste text:

Button 1 Down	Start Selection
Button 1 Motion	Adjust Selection
Button 1 Up	End Selection (cut)
Button 2 Down	Insert Current Selection (paste)
Button 3 Down	Extend Current Selection
Button 3 Motion	Adjust Selection
Button 3 Up	End Selection (cut)

CONFIRMATION DIALOG BOXES

Whenever you press a button that may cause you to lose some work or is otherwise dangerous, a popup dialog box will appear asking you to confirm the action. This window will contain an ''Abort'' or ''No'' button and a ''Confirm'' or ''Yes'' button. Pressing the ''No'' button cancels the operation, and pressing the ''Yes'' will proceed with the operation.

Some dialog boxes contain messages from *MH*. Clicking on the message field will cause the dialog box to resize so that you can read the entire message.

MESSAGE-SEQUENCES

An *MH* message sequence is just a set of messages associated with some name. They are local to a particular folder; two different folders can have sequences with the same name. In all folders, the sequence ''all'' is predefined; it consists of the set of all messages in that folder.. As many as nine sequences may be defined for each folder, including the predefined ''all'' sequence. (The sequence ''cur'' is also usually defined for every folder; it consists of only the current message. *xmh* hides ''cur'' from the user, instead placing a ''+'' by the current message. Also, *xmh* does not support the ''unseen'' sequence, so that one is also hidden from the user.)

The message sequences for a folder (including one for ''all'') are displayed in the ''Sequence'' menu, below the sequence commands. The table of contents (also known as the ''toc'') is at any one time displaying one message sequence. This is called the ''viewed sequence'', and its name will be displayed in the toc title bar just after the folder name. Also, at any time one of the sequences in the menu will have a check mark next to it. This is called the ''selected sequence''. Note that the viewed sequence and the selected sequence are not necessarily the same. (This all pretty much corresponds to the way the folders work.)

The **Open Sequence, Add to Sequence, Remove from Sequence,** and **Delete Sequence** commands are active only if the viewed folder contains message-sequences.

Note that none of the above actually affect whether a message is in the folder. Remember that a sequence is a set of messages within the folder; the above operations just affect what messages are in that set.

To create a new sequence, select the ''Pick'' menu entry. A new window will appear, with lots of places to enter text. Basically, you can describe the sequence's initial set of messages based on characteristics of the message. Thus, you can define a sequence to be all the messages that were from a particular person, or with a particular subject, and so on. You can also connect things up with boolean operators, so you can select all things from ''weissman'' with the subject ''xmh''.

Hopefully, the layout is fairly obvious. The simplest cases are the easiest: just point to the proper field and type. If you enter in more than one field, it will only select messages which match all non-empty fields.

The more complicated cases arise when you want things that match one field or another one, but not necessarily both. That's what all the ''or'' buttons are for. If you want all things with the subject ''xmh'' or ''xterm'', just press the ''or'' button next to the ''Subject:'' field. Another box will appear where you can enter another subject.

If you want all things either from ''weissman'' or with subject ''xmh'', but not necessarily both, select the ''-Or-'' button. This will essentially double the size of the form. You can then enter ''weissman'' in a from: box on the top half, and ''xmh'' in a subject: box on the lower part.

If you select the ''Skip'' button, then only those messages that *don't* match the fields on that row are included.

Finally, in the bottom part of the window will appear several more boxes. One is the name of the sequence you're defining. (It defaults to the name of the selected sequence when ''Pick'' was pressed, or to ''temp'' if ''all'' was the selected sequence.) Another box defines which sequence to look through for potential members of this sequence; it defaults to the viewed sequence when ''Pick'' was pressed.

Two more boxes define a date range; only messages within that date range will be considered. These dates must be entered in 822-style format: each date is of the form ''dd mmm yy hh:mm:ss zzz'', where dd is a one or two digit day of the month, mmm is the three-letter abbreviation for a month, and yy is a year. The remaining fields are optional: hh, mm, and ss specify a time of day, and zzz selects a time zone. Note that if the time is left out, it defaults to midnight; thus if you select a range of ''7 nov 86'' - ''8 nov 86'', you will only get messages from the 7th, as all messages on the 8th will have arrived after midnight.

''Date field'' specifies which date field in the header to look at for this date range; it probably won't be useful to anyone. If the sequence you're defining already exists, you can optionally merge the old set with the new; that's what the ''Yes'' and ''No'' buttons are all about. Finally, you can ''OK'' the whole thing, or ''Cancel'' it.

In general, most people will rarely use these features. However, it's nice to occasionally use ''Pick'' to find some messages, look through them, and then hit ''Delete Sequence'' to put things back in their original state.

WIDGET HIERARCHY

In order to specify resources, it is useful to know the hierarchy of widgets which compose *xmh*. In the notation below, indentation indicates hierarchical structure. The widget class name is given first, followed by the widget instance name. The application class name is Xmh.

The hierarchy of the main toc and view window is identical for additional toc and view windows, except that a topLevelShell widget is inserted in the hierarchy between the application shell and the Paned widget.

```
Xmh xmh
        Paned xmh
                SimpleMenu  folderMenu
                        SmeBSB  open
                        SmeBSB  openInNew
                        SmeBSB  create
                        SmeBSB  delete
                        SmeLine  line
                        SmeBSB  close
                SimpleMenu  tocMenu
                        SmeBSB  inc
                        SmeBSB  commit
                        SmeBSB  pack
                        SmeBSB  sort
                        SmeBSB  rescan
```

```
SimpleMenu  messageMenu
          SmeBSB  compose
          SmeBSB  next
          SmeBSB  prev
          SmeBSB  delete
          SmeBSB  move
          SmeBSB  copy
          SmeBSB  unmark
          SmeBSB  viewNew
          SmeBSB  reply
          SmeBSB  forward
          SmeBSB  useAsComp
          SmeBSB  print
SimpleMenu  sequenceMenu
          SmeBSB  pick
          SmeBSB  openSeq
          SmeBSB  addToSeq
          SmeBSB  removeFromSeq
          SmeBSB  deleteSeq
          SmeLine  line
          SmeBSB  all
SimpleMenu  viewMenu
          SmeBSB  reply
          SmeBSB  forward
          SmeBSB  useAsComp
          SmeBSB  edit
          SmeBSB  save
          SmeBSB  print
SimpleMenu  optionMenu
          SmeBSB  reverse
Viewport.Core  menuBox.clip
          Box  menuBox
                    MenuButton  folderButton
                    MenuButton  tocButton
                    MenuButton  messageButton
                    MenuButton  sequenceButton
                    MenuButton  viewButton
                    MenuButton  optionButton
Grip  grip
Label folderTitlebar
Grip  grip
Viewport.Core  folders.clip
          Box  folders
```

```
                        MenuButton  inbox
                        MenuButton  drafts
                                SimpleMenu  menu
                                        SmeBSB <folder_name>
                                              .
                                              .
                                              .

                Grip  grip
                Label  tocTitlebar
                Grip  grip
                Text toc
                        Scrollbar  vScrollbar
                Grip  grip
                Label  viewTitlebar
                Grip  grip
                Text  view
                        Scrollbar  vScrollbar
                        Scrollbar  hScrollbar
```

The hierarchy of the Create Folder popup dialog box:

```
        transientShell  prompt
                Dialog  dialog
                        Label  label
                        Text  value
                        Command  okay
                        Command  cancel
```

The hierarchy of the Notice dialog box, which reports messages from MH:

```
        transientShell  notice
                Dialog  dialog
                        Label  label
                        Text  value
                        Command  confirm
```

The hierarchy of the Confirmation dialog box:

```
        transientShell  confirm
                Dialog  dialog
                        Label  label
                        Command  yes
```

```
                    Command  no
```

The hierarchy of the dialog box which reports errors:

```
transientShell  error
        Dialog  dialog
                Label  label
                Command  OK
```

The hierarchy of the composition window:

```
topLevelShell  xmh
        Paned  xmh
                Label  composeTitlebar
                Text  comp
                Viewport.Core  compButtons.clip
                        Box  compButtons
                                Command  close
                                Command  send
                                Command  reset
                                Command  compose
                                Command  save
                                Command  insert
```

The hierarchy of the view window:

```
topLevelShell  xmh
        Paned  xmh
                Label  viewTitlebar
                Text  view
                Viewport.Core  viewButtons.clip
                        Box  viewButtons
                                Command  close
                                Command  reply
                                Command  forward
                                Command  useAsComp
                                Command  edit
                                Command  save
                                Command  print
```

The hierarchy of the pick window:
(Unnamed widgets have no name.)

```
          topLevelShell  xmh
                Paned  xmh
                      Label  pickTitlebar
                      Viewport.core  pick.clip
                            Form  form
                                  Form
      The first 6 rows of the pick window have identical structure:
                                        Form
                                              Toggle
                                              Toggle
                                              Label
                                              Text
                                              Command

                                        Form
                                              Toggle
                                              Toggle
                                              Text
                                              Text
                                              Command
                                        Form
                                              Command
                      Viewport.core  pick.clip
                            Form  form
                                  From
                                        Form
                                              Label
                                              Text
                                              Label
                                              Text
                                        Form
                                              Label
                                              Text
                                              Label
                                              Text
                                              Label
                                              Text
                                        Form
                                              Label
                                              Toggle
                                              Toggle
                                        Form
                                              Command
```

Command

APPLICATION-SPECIFIC RESOURCES

Resource instance names begin with a lower case letter but are otherwise identical to the class name.

If TocGeometry, ViewGeometry, CompGeometry, or PickGeometry are not specified, then the value of Geometry is used instead. If the resulting height is not specified (e.g., "", "=500", "+0-0"), then the default height of windows is calculated from fonts and line counts. If the width is not specified (e.g., "", "=x300", "-0+0), then half of the display width is used. If unspecified, the height of a pick window defaults to half the height of the display.

Any of these options may also be specified on the command line by using the X Toolkit Intrinsics resource specification mechanism. Thus, to run *xmh* showing all message headers,

```
% xmh -xrm '*HideBoringHeaders:off'
```

The following resources are defined:

Banner A short string that is the default label of the folder, Table of Contents, and view. The default is "xmh MIT X Consortium R4".

BlockEventsOnBusy
Whether to disallow user input and show a busy cursor while *xmh* is busy processing a command. Default is true.

BusyCursor
The name of the symbol used to represent the position of the pointer, displayed if **BlockEventsOnBusy** is true, when *xmh* is processing a time-consuming command. The default is "watch".

BusyPointerColor
The foreground color of the busy cursor. Default is XtDefault-Foreground.

CheckFrequency
How often to check for new mail, make checkpoints, and rescan the Table of Contents, in minutes. If **CheckNewMail** is true, *xmh* checks to see if you have new mail each interval. If **MakeCheckpoints** is true, checkpoints are made every fifth interval. Also every fifth interval, the Table of Contents is checked for inconsistencies with the file system, and rescanned. To prevent all of these checks from occurring, set **CheckFrequency** to 0. The default is 1.

CheckNewMail
If true, *xmh* will check at regular intervals to see if new mail has arrived for any of the folders. A visual indication will be given if new mail is waiting to be retrieved. Default is True. (See BUGS). The interval can be adjusted with the **CheckFrequency**.

CommandButtonCount
The number of command buttons to create in a button box in between the toc and the view areas of the main window. *xmh* will create these buttons with the names *button1, button2* and so on, in a box with the name *commandBox*. The user can specify labels and actions for the buttons in a private resource file; see the section on Actions. The default is 0.

CompGeometry
Initial geometry for windows containing compositions.

Cursor The name of the symbol used to represent the pointer. Default is ''left_ptr''.

DraftsFolder
The folder used for message drafts. Default is ''drafts''.

Geometry
Default geometry to use. Default is none.

HideBoringHeaders
If ''on'', then *xmh* will attempt to skip uninteresting header lines within messages by scrolling them off. Default is ''on''.

InitialFolder
Which folder to display on startup. May also be set with the command-line option **-initial**. Default is ''inbox''.

InitialIncFile
The file name of your incoming mail drop. *xmh* tries to construct a filename for the ''inc -file'' command, but in some installations (e.g. those using the Post Office Protocol) no file is appropriate. In this case, **InitialIncFile** should be specified as the empty string, and *inc* will be invoked without a -file argument. The default is to use the value of the environment variable **MAIL**, or if that is not set, to append the value of the environment variable **USER** to */usr/spool/mail/*.

MailPath
The full path prefix for locating your mail folders. May also be set with the command-line option, **-path**. The default is the Path component in $HOME/.mh_profile, or ''$HOME/Mail'' if none.

MailWaitingFlag
If true, *xmh* will attempt to set an indication in its icon when new mail is waiting to be retrieved. If this option is true, then CheckNewMail is assumed to be true as well. The **-flag** command line option is a quick way to turn MailWaitingFlag on.

MakeCheckpoints
If true, *xmh* will attempt to save checkpoints of volatile information. The frequency of checkpointing is controlled by the resource **CheckFrequency**.

MhPath What directory in which to find the *MH* commands. If a command isn't found here, then the directories in the user's path are searched. Default is ''/usr/local/mh6''.

PickGeometry
Initial geometry for pick windows.

PointerColor
The foreground color of the pointer. Default is XtDefaultForeground.

PrefixWmAndIconName
Whether to prefix the window and icon name with "xmh: ". Default is true.

PrintCommand
What sh command to execute to print a message. Note that stdout and stderr must be specifically redirected! If a message or range of messages is selected for printing, the full file paths of each message file is appended to the specified print command. The default is ''enscript >/dev/null 2>/dev/null''.

ReplyInsertFilter
A shell command to be executed when the *Insert* button is activated in a composition window. The full path and filename of the source message is added to the end of the command before being passed to *sh*(1). The default filter is *cat*; i.e. it inserts the entire message into the composition. Interesting filters are: *awk -e '{print "   " $0}'* or *<mh directory>/lib/mhl -form mhl.body.*

ReverseReadOrder
When true, the next message will be the message prior to the current message in the table of contents, and the previous message will be the message after the current message in the table of contents. The default is false.

SendBreakWidth

When a message is sent from *xmh*, lines longer than this value will be split into multiple lines, each of which is no longer than **SendWidth**. This value may be overridden for a single message by inserting an additional line in the message header of the form *SendBreakWidth: value*. This line will be removed from the header before the message is sent. The default is 85.

SendWidth

When a message is sent from *xmh*, lines longer than **SendBreakWidth** characters will be split into multiple lines, each of which is no longer than this value. This value may be overridden for a single message by inserting an additional line in the message header of the form *SendWidth: value*. This line will be removed from the header before the message is sent. The default is 72.

SkipCopied

Whether to skip over messages marked for copying when using "View Next Message" and "View Previous Message". Default is true.

SkipDeleted

Whether to skip over messages marked for deletion when using "View Next Message" and "View Previous Message". Default is true.

SkipMoved

Whether to skip over messages marked for moving to other folders when using "View Next Message" and "View Previous Message". Default is true.

StickyMenu

If true, when popup command menus are used, the most recently selected entry will be under the cursor when the menu pops up. Default is false. See the file *clients/xmh/Xmh.sample* for an example of how to specify resources for pop up command menus.

TempDir

Directory for *xmh* to store temporary directories. For privacy, a user might want to change this to a private directory. Default is "/tmp".

TocGeometry

Initial geometry for master *xmh* windows.

TocPercentage

The percentage of the main window that is used to display the Table of Contents. Default is 33.

TocWidth

How many characters to generate for each message in a folder's table of contents. Default is 100. Use 80 if you plan to use *mhl* a lot, because it will be faster, and the extra 20 characters may not be useful.

ViewGeometry

Initial geometry for windows showing only a view of a message.

ACTIONS

Because *xmh* provides action procedures which correspond to command functionality and installs accelerators, users can customize accelerators in a private resource file. *xmh* provides action procedures which correspond to entries in the command menus; these are given in the sections describing menu commmands. For examples of specifying customized resources, see the file *clients/xmh/Xmh.sample*. Unpredictable results can occur if actions are bound to events or widgets for which they were not designed.

In addition to the actions corresponding to commands, these action routines are defined:

XmhPushFolder([*foldername, ...*])

This action pushes each of its argument(s) onto a stack of foldernames. If no arguments are given, the selected folder is pushed onto the stack.

XmhPopFolder()

This action pops one foldername from the stack and sets the selected folder.

XmhPopupFolderMenu()

This action should always be taken when the user selects a folder button. A folder button represents a folder and zero or more subfolders. The menu of subfolders is built upon the first reference, by this routine. If there are no subfolders, this routine will mark the folder as having no subfolders, and no menu will be built. In that case the menu button emulates a toggle button. When subfolders exist, the menu will popup, using the menu button action PopupMenu().

XmhSetCurrentFolder()
This action allows menu buttons to emulate toggle buttons in the function of selecting a folder. This action is for menu button widgets only, and sets the selected folder.

XmhLeaveFolderButton()
This action insures that the menu button behaves properly when the user moves the pointer out of the menu button window.

XmhPushSequence([*sequencename*, ...]**)**
This action pushes each of its arguments onto the stack of sequence names. If no arguments are given, the selected sequence is pushed onto the stack.

XmhPopSequence() This action pops one sequence name from the stack of sequence names, which then becomes the selected sequence.

XmhPromptOkayAction()
This action is equivalent to pressing the okay button in the Create Folder popup.

XmhCancelPick() This action is equivalent to pressing the cancel button in the pick window.

CUSTOMIZATION USING *MH*

The initial text displayed in a composition window is generated by executing the corresponding *MH* command; i.e. *comp*, *repl*, or *forw*, and therefore message components may be customized as specified for those commands. *Comp* is executed only once per invocation of *xmh* and the message template is re-used for each successive new composition.

FILES

~/Mail
~/.mh_profile - *MH* profile
/usr/local/mh6 - *MH* commands
~/Mail/<folder>/.xmhcache - scan folder
~/Mail/<folder>/.mh_sequences - sequence definitions
/tmp - temporary files

SEE ALSO

X(1), xrdb(1), X Toolkit Intrinsics, Athena Widget Set, mh(1), enscript(1)

BUGS

- Printing support is minimal.
- Should handle the ``unseen'' message-sequence.
- Should determine by itself if the user hasn't used *MH* before, and offer to create the .mh_profile, instead of hanging on inc.
- Still a few commands missing (rename folder, remail message).
- A bug in *MH* limits the the number of characters in .mh_sequences to BUFSIZ. When the limit is reached, the .mh_sequences file often becomes corrupted, and sequence definitions may be lost.
- Except for the icon, there isn't an indication that you have new mail.
- There should be a resource, ShowOnInc, which when true, would show the current message in the view after incorporating new mail.
- The CheckFrequency resource should be split into two separate resources.
- WM_SAVE_YOURSELF protocol is ignored.
- WM_DELETE_WINDOW protocol doesn't work right when requesting deletion of the first toc and view, while trying to keep other *xmh* windows around.
- Doesn't support annotations when replying to messages.

COPYRIGHT

Copyright 1988, 1989, Digital Equipment Corporation.
Copyright 1989, Massachusetts Institute of Technology
See *X(1)* for a full statement of rights and permissions.

AUTHOR

Terry Weissman, Digital Western Research Laboratory
modified by Donna Converse, MIT X Consortium

NAME

xmodmap - utility for modifying keymaps in X

SYNOPSIS

xmodmap [-options ...] [filename]

DESCRIPTION

The *xmodmap* program is used to edit and display the keyboard *modifier map* and *keymap table* that are used by client applications to convert event keycodes into keysyms. It is usually run from the user's session startup script to configure the keyboard according to personal tastes.

OPTIONS

The following options may be used with *xmodmap*:

−display *display*
This option specifies the host and display to use.

−help This option indicates that a brief description of the command line arguments should be printed on the standard error. This will be done whenever an unhandled argument is given to *xmodmap*.

−grammar
This option indicates that a help message describing the expression grammar used in files and with -e expressions should be printed on the standard error.

−verbose
This option indicates that *xmodmap* should print logging information as it parses its input.

−quiet This option turns off the verbose logging. This is the default.

−n This option indicates that *xmodmap* should not change the mappings, but should display what it would do, like *make(1)* does when given this option.

−e *expression*
This option specifies an expression to be executed. Any number of expressions may be specified from the command line.

−pm This option indicates that the current modifier map should be printed on the standard output.

−pk This option indicates that the current keymap table should be printed on the standard output.

−pp This option indicates that the current pointer map should be printed on the standard output.

– A lone dash means that the standard input should be used as the input file.

The *filename* specifies a file containing *xmodmap* expressions to be executed. This file is usually kept in the user's home directory with a name like *.xmodmaprc*.

EXPRESSION GRAMMAR

The *xmodmap* program reads a list of expressions and parses them all before attempting execute any of them. This makes it possible to refer to keysyms that are being redefined in a natural way without having to worry as much about name conflicts.

keycode *NUMBER = KEYSYMNAME ...*
: The list of keysyms is assigned to the indicated keycode (which may be specified in decimal, hex or octal and can be determined by running the *xev* program in the examples directory). Usually only one keysym is assigned to a given code.

keysym *KEYSYMNAME = KEYSYMNAME ...*
: The *KEYSYMNAME* on the left hand side is looked up to find its current keycode and the line is replaced with the appropriate **keycode** expression. Note that if you have the same keysym bound to multiple keys, this might not work.

clear *MODIFIERNAME*
: This removes all entries in the modifier map for the given modifier, where valid name are: Shift, Lock, Control, Mod1, Mod2, Mod3, Mod4 and Mod5 (case does not matter in modifier names, although it does matter for all other names). For example, ''clear Lock'' will remove all any keys that were bound to the shift lock modifier.

add *MODIFIERNAME = KEYSYMNAME ...*
: This adds the given keysyms to the indicated modifier map. The keysym names are evaluated after all input expressions are read to make it easy to write expressions to swap keys (see the EXAMPLES section).

remove *MODIFIERNAME = KEYSYMNAME ...*
: This removes the given keysyms from the indicated modifier map. Unlike **add,** the keysym names are evaluated as the line is read in. This allows you to remove keys from a modifier without having to worry about whether or not they have been reassigned.

pointer = default
: This sets the pointer map back to its default settings (button 1 generates a code of 1, button 2 generates a 2, etc.).

pointer = *NUMBER ...*
: This sets to pointer map to contain the indicated button codes. The list always starts with the first physical button.

Lines that begin with an exclamation point (!) are taken as comments.

If you want to change the binding of a modifier key, you must also remove it from the appropriate modifier map.

EXAMPLES

Many pointers are designed such the first button is pressed using the index finger of the right hand. People who are left-handed frequently find that it is more comfortable to reverse the button codes that get generated so that the primary button is pressed using the index finger of the left hand. This could be done on a 3 button pointer as follows:

```
% xmodmap -e "pointer = 3 2 1"
```

Many editor applications support the notion of Meta keys (similar to Control keys except that Meta is held down instead of Control). However, some servers do not have a Meta keysym in the default keymap table, so one needs to be added by hand. The following command will attach Meta to the Multi-language key (sometimes label Compose Character). It also takes advantage of the fact that applications that need a Meta key simply need to get the keycode and don't require the keysym to be in the first column of the keymap table. This means that applications that are looking for a Multi_key (including the default modifier map) won't notice any change.

```
% keysym Multi_key = Multi_key Meta_L
```

One of the more simple, yet convenient, uses of *xmodmap* is to set the keyboard's "rubout" key to generate an alternate keysym. This frequently involves exchanging Backspace with Delete to be more comfortable to the user. If the *ttyModes* resource in *xterm* is set as well, all terminal emulator windows will use the same key for erasing characters:

```
% xmodmap -e "keysym BackSpace = Delete"
% echo "XTerm*ttyModes: erase ^?" | xrdb -merge
```

Some keyboards do not automatically generate less than and greater than characters when the comma and period keys are shifted. This can be remedied with *xmodmap* by resetting the bindings for the comma and period with the following scripts:

```
!
! make shift-, be < and shift-. be >
!
keysym comma = comma less
keysym period = period greater
```

One of the more irritating differences between keyboards is the location of the Control and Shift Lock keys. A common use of *xmodmap* is to swap these two keys as follows:

```
!
! Swap Caps_Lock and Control_L
!
remove Lock = Caps_Lock
remove Control = Control_L
keysym Control_L = Caps_Lock
keysym Caps_Lock = Control_L
add Lock = Caps_Lock
add Control = Control_L
```

The *keycode* command is useful for assigning the same keysym to multiple keycodes. Although unportable, it also makes it possible to write scripts that can reset the keyboard to a known state. The following script sets the backspace key to generate Delete (as shown above), flushes all existing caps lock bindings, makes the CapsLock key be a control key, make F5 generate Escape, and makes Break/Reset be a shift lock.

```
!
! On the HP, the following keycodes have key caps as listed:
!
!    101  Backspace
!     55  Caps
!     14  Ctrl
!     15  Break/Reset
!     86  Stop
!     89  F5
!
keycode 101 = Delete
keycode 55 = Control_R
clear Lock
add Control = Control_R
keycode 89 = Escape
keycode 15 = Caps_Lock
```

```
        add Lock = Caps_Lock
```

ENVIRONMENT

DISPLAY

to get default host and display number.

SEE ALSO

X(1)

BUGS

Every time a **keycode** expression is evaluated, the server generates a *MappingNotify* event on every client. This can cause some thrashing. All of the changes should be batched together and done at once. Clients that receive keyboard input and ignore *MappingNotify* events will not notice any changes made to keyboard mappings.

Xmodmap should generate "add" and "remove" expressions automatically whenever a keycode that is already bound to a modifier is changed.

There should be a way to have the *remove* expression accept keycodes as well as keysyms for those times when you really mess up your mappings.

COPYRIGHT

AUTHOR

Jim Fulton, MIT X Consortium, rewritten from an earlier version by David Rosenthal of Sun Microsystems.

NAME

xpr – print an X window dump

SYNOPSIS

xpr [**–device** *dev*] [**–scale** *scale*] [**–height** *inches*] [**–width** *inches*] [**–left** *inches*] [**–top** *inches*] [**–header** *string*] [**–trailer** *string*] [**–landscape**] [**–portrait**] [**–plane** *number*] [**–gray**] [**–rv**] [**–compact**] [**–output** *filename*] [**–append** *filename*] [**–noff**] [**–split** *n*] [**–psfig**] [**–density** *dpi*] [**–cutoff** *level*] [**–noposition**] [**–gamma** *correction*] [**–render** *algorithm*] [**–slide**] [*filename*]

DESCRIPTION

xpr takes as input a window dump file produced by *xwd(1)* and formats it for output on PostScript printers, the Digital LN03 or LA100, the IBM PP3812 page printer, the HP LaserJet (or other PCL printers), or the HP PaintJet. If no file argument is given, the standard input is used. By default, *xpr* prints the largest possible representation of the window on the output page. Options allow the user to add headers and trailers, specify margins, adjust the scale and orientation, and append multiple window dumps to a single output file. Output is to standard output unless **–output** is specified.

Command Options

–device *dev*

Specifies the device on which the file will be printed. Currently supported:

la100	Digital LA100
ljet	HP LaserJet series and other monochrome PCL devices such as ThinkJet, QuietJet, RuggedWriter, HP2560 series, and HP2930 series printers
ln03	Digital LN03
pjet	HP PaintJet (color mode)
pjetxl	HP HP PaintJet XL Color Graphics Printer (color mode)
pp	IBM PP3812
ps	PostScript printer

The default device is the *LN03*, for historical reasons. **-device lw** (LaserWriter) is equivalent to -device ps and is provided only for backwards compatibility.

–scale *scale*

Affects the size of the window on the page. The PostScript, LN03, and HP printers are able to translate each bit in a window pixel map into a grid of a specified size. For example each bit might translate into a 3x3 grid. This would be specified by **–scale** *3*. By default a window is printed with the largest scale that will fit onto the page for the specified orientation.

–height *inches*
Specifies the maximum height of the page.

–width *inches*
Specifies the maximum width of the page.

–left *inches*
Specifies the left margin in inches. Fractions are allowed. By default the window is centered in the page.

–top *inches*
Specifies the top margin for the picture in inches. Fractions are allowed.

–header *string*
Specifies a header string to be printed above the window.

–trailer *string*
Specifies a trailer string to be printed below the window.

–landscape
Forces the window to printed in landscape mode. By default a window is printed such that its longest side follows the long side of the paper.

–plane *number*
Specifies which bit plane to use in an image. The default is to use the entire image and map values into black and white based on color intensities.

–gray Uses a simple 5-level gray scale conversion on a color image, rather than mapping to strictly black and white. This essentially doubles the effective width and height of the image.

–portrait
Forces the window to be printed in portrait mode. By default a window is printed such that its longest side follows the long side of the paper.

–rv Forces the window to be printed in reverse video.

–compact
Uses simple run-length encoding for compact representation of windows with lots of white pixels.

–output *filename*
: Specifies an output file name. If this option is not specified, standard output is used.

–append *filename*
: Specifies a filename previously produced by *xpr* to which the window is to be appended.

–noff When specified in conjunction with **–append**, the window will appear on the same page as the previous window.

–split *n* This option allows the user to split a window onto several pages. This might be necessary for very large windows that would otherwise cause the printer to overload and print the page in an obscure manner.

–psfig Suppress translation of the PostScript picture to the center of the page.

–density *dpi*
: Indicates what dot-per-inch density should be used by the HP printer.

–cutoff *level*
: Changes the intensity level where colors are mapped to either black or white for monochrome output on a LaserJet printer. The *level* is expressed as percentage of full brightness. Fractions are allowed.

–noposition
: This option causes header, trailer, and image positioning command generation to be bypassed for LaserJet, PaintJet and PaintJet XL printers.

–gamma *correction*
: This changes the intensity of the colors printed by PaintJet XL printer. The *correction* is a floating point value in the range 0.00 to 3.00. Consult the operator's manual to determine the correct value for the specific printer.

–render *algorithm*
: This allows PaintJet XL printer to render the image with the best quality versus performance tradeoff. Consult the operator's manual to determine which *algorithms* are available.

–slide This option allows overhead transparencies to be printed using the PaintJet and PaintJet XL printers.

SEE ALSO

xwd(1), xwud(1), X(1)

LIMITATIONS

The current version of *xpr* can generally print out on the LN03 most X windows that are not larger than two-thirds of the screen. For example, it will be able to print out a large Emacs window, but it will usually fail when trying to print out the entire screen. The LN03 has memory limitations that can cause it to incorrectly print very large or complex windows. The two most common errors encountered are ''band too complex'' and ''page memory exceeded.'' In the first case, a window may have a particular six pixel row that contains too many changes (from black to white to black). This will cause the printer to drop part of the line and possibly parts of the rest of the page. The printer will flash the number '1' on its front panel when this problem occurs. A possible solution to this problem is to increase the scale of the picture, or to split the picture onto two or more pages. The second problem, ''page memory exceeded,'' will occur if the picture contains too much black, or if the picture contains complex half-tones such as the background color of a display. When this problem occurs the printer will automatically split the picture into two or more pages. It may flash the number '5' on its from panel. There is no easy solution to this problem. It will probably be necessary to either cut and paste, or to rework the application to produce a less complex picture.

There are several limitations on the LA100 support: the picture will always be printed in portrait mode, there is no scaling, and the aspect ratio will be slightly off.

Support for PostScript output currently cannot handle the **-append**, **-noff** or **-split** options.

The **-compact** option is *only* supported for PostScript output. It compresses white space but not black space, so it is not useful for reverse-video windows.

For color images, should map directly to PostScript image support.

HP PRINTERS

If no –**density** is specified on the command line 300 dots per inch will be assumed for *ljet* and 90 dots per inch for *pjet*. Allowable *density* values for a LaserJet printer are 300, 150, 100, and 75 dots per inch. Consult the operator's manual to determine densities supported by other printers.

If no –**scale** is specified the image will be expanded to fit the printable page area.

The default printable page area is 8x10.5 inches. Other paper sizes can be accomodated using the **–height** and **–width** options.

Note that a 1024x768 image fits the default printable area when processed at 100 dpi with scale=1, the same image can also be printed using 300 dpi with scale=3 but will require considerably more data be transfered to the printer.

xpr may be tailored for use with monochrome PCL printers other than the LaserJet. To print on a ThinkJet (HP2225A) *xpr* could be invoked as:

```
xpr -density 96 -width 6.667 filename
```

or for black-and-white output to a PaintJet:

```
xpr -density 180 filename
```

The monochrome intensity of a pixel is computed as 0.30*R + 0.59*G + 0.11*B. If a pixel's computed intensity is less than the **–cutoff** level it will print as white. This maps light-on-dark display images to black-on-white hardcopy. The default cutoff intensity is 50% of full brightness. Example: specifying **–cutoff 87.5** moves the white/black intensity point to 87.5% of full brightness.

A LaserJet printer must be configured with sufficient memory to handle the image. For a full page at 300 dots per inch approximately 2MB of printer memory is required.

Color images are produced on the PaintJet at 90 dots per inch. The PaintJet is limited to sixteen colors from its 330 color palette on each horizontal print line. *xpr* will issue a warning message if more than sixteen colors are encountered on a line. *xpr* will program the PaintJet for the first sixteen colors encountered on each line and use the nearest matching programmed value for other colors present on the line.

Specifying the **–rv**, reverse video, option for the PaintJet will cause black and white to be interchanged on the output image. No other colors are changed.

Multiplane images must be recorded by *xwd* in *ZPixmap* format. Single plane (monochrome) images may be in either *XYPixmap* or *ZPixmap* format.

Some PCL printers do not recognize image positioning commands. Output for these printers will not be centered on the page and header and trailer strings may not appear where expected.

The –**gamma** and **-render** options are supported only on the PaintJet XL printers.

The –**slide** option is not supported for LaserJet printers.

The –**split** option is not supported for HP printers.

COPYRIGHT

Copyright 1988, Massachusetts Institute of Technology.
Copyright 1986, Marvin Solomon and the University of Wisconsin.
Copyright 1988, Hewlett Packard Company.
See *X(1)* for a full statement of rights and permissions.

AUTHORS

Michael R. Gretzinger, MIT Project Athena, Jose Capo, MIT Project Athena (PP3812 support), Marvin Solomon, University of Wisconsin, Bob Scheifler, MIT, Angela Bock and E. Mike Durbin, Rich Inc. (grayscale), Larry Rupp, HP (HP printer support).

NAME

xprop - property displayer for X

SYNOPSIS

xprop [-help] [-grammar] [-id *id*] [-root] [-name *name*] [-frame] [-font *font*] [-display *display*] [-len *n*] [-notype] [-fs *file*] [-f *atom format* [*dformat*]]* [*format* [*dformat*] *atom*]*

SUMMARY

The *prop* utility is for displaying window and font properties in an X server. One window or font is selected using the command line arguments or possibly in the case of a window, by clicking on the desired window. A list of properties is then given, possibly with formatting information.

OPTIONS

-help Print out a summary of command line options.

-grammar
Print out a detailed grammar for all command line options.

-id *id* This argument allows the user to select window *id* on the command line rather than using the pointer to select the target window. This is very useful in debugging X applications where the target window is not mapped to the screen or where the use of the pointer might be impossible or interfere with the application.

-name *name*
This argument allows the user to specify that the window named *name* is the target window on the command line rather than using the pointer to select the target window.

-font*font*
This argument allows the user to specify that the properties of font*font* should be displayed.

-root This argument specifies that X's root window is the target window. This is useful in situations where the root window is completely obscured.

-display *display*
This argument allows you to specify the server to connect to; see *X(1)*.

-len *n* Specifies that at most *n* bytes of any property should be read or displayed.

-notype Specifies that the type of each property should not be displayed.

-fs *file* Specifies that file *file* should be used as a source of more formats for properties.

-frame Specifies that when selecting a window by hand (i.e. if none of **-name**, **-root**, or **-id** are given), look at the window manager frame (if any) instead of looking for the client window.

-remove *property-name*
Specifies the name of a property to be removed from the indicated window.

-f *name format* [*dformat*]
Specifies that the *format* for *name* should be *format* and that the *dformat* for *name* should be *dformat*. If *dformat* is missing, " = $0+\n" is assumed.

DESCRIPTION

For each of these properties, its value on the selected window or font is printed using the supplied formatting information if any. If no formatting information is supplied, internal defaults are used. If a property is not defined on the selected window or font, "not defined" is printed as the value for that property. If no property list is given, all the properties possessed by the selected window or font are printed.

A window may be selected in one of four ways. First, if the desired window is the root window, the -root argument may be used. If the desired window is not the root window, it may be selected in two ways on the command line, either by id number such as might be obtained from *xwininfo*, or by name if the window possesses a name. The -id argument selects a window by id number in either decimal or hex (must start with 0x) while the -name argument selects a window by name.

The last way to select a window does not involve the command line at all. If none of -font, -id, -name, and -root are specified, a crosshairs cursor is displayed and the user is allowed to choose any visible window by pressing any pointer button in the desired window. If it is desired to display properties of a font as opposed to a window, the -font argument must be used.

Other than the above four arguments and the -help argument for obtaining help, and the -grammar argument for listing the full grammar for the command line, all the other command line arguments are used in specifing both the format of the properties to be displayed and how to display them. The -len *n* argument specifies that at most *n* bytes of any given property will be read and displayed. This is useful for example when displaying the cut buffer on the root window which could run to several pages if displayed in full.

Normally each property name is displayed by printing first the property name then its type (if it has one) in parentheses followed by its value. The -notype argument specifies that property types should not be displayed. The -fs argument is used to specify a file containing a list of formats for properties while the -f argument is used to specify the format for one property.

The formatting information for a property actually consists of two parts, a *format* and a *dformat*. The *format* specifies the actual formatting of the property (i.e., is it made up of words, bytes, or longs?, etc.) while the *dformat* specifies how the property should be displayed.

The following paragraphs describe how to construct *format*s and *dformat*s. However, for the vast majority of users and uses, this should not be necessary as the built in defaults contain the *format*s and *dformat*s necessary to display all the standard properties. It should only be necessary to specify *format*s and *dformat*s if a new property is being dealt with or the user dislikes the standard display format. New users especially are encouraged to skip this part.

A *format* consists of one of 0, 8, 16, or 32 followed by a sequence of one or more format characters. The 0, 8, 16, or 32 specifies how many bits per field there are in the property. Zero is a special case meaning use the field size information associated with the property itself. (This is only needed for special cases like type INTEGER which is actually three different types depending on the size of the fields of the property)

A value of 8 means that the property is a sequence of bytes while a value of 16 would mean that the property is a sequence of words. The difference between these two lies in the fact that the sequence of words will be byte swapped while the sequence of bytes will not be when read by a machine of the opposite byte order of the machine that orginally wrote the property. For more information on how properties are formatted and stored, consult the Xlib manual.

Once the size of the fields has been specified, it is necessary to specify the type of each field (i.e., is it an integer, a string, an atom, or what?) This is done using one format character per field. If there are more fields in the property than format characters supplied, the last character will be repeated as many times as necessary for the extra fields. The format characters and their meaning are as follows:

a	The field holds an atom number. A field of this type should be of size 32.
b	The field is an boolean. A 0 means false while anything else means true.

c	The field is an unsigned number, a cardinal.
i	The field is a signed integer.
m	The field is a set of bit flags, 1 meaning on.
s	This field and the next ones until either a 0 or the end of the property represent a sequence of bytes. This format character is only usable with a field size of 8 and is most often used to represent a string.
x	The field is a hex number (like 'c' but displayed in hex - most useful for displaying window ids and the like)

An example *format* is 32ica which is the format for a property of three fields of 32 bits each, the first holding a signed integer, the second an unsigned integer, and the third an atom.

The format of a *dformat* unlike that of a *format* is not so rigid. The only limitations on a *dformat* is that one may not start with a letter or a dash. This is so that it can be distingished from a property name or an argument. A *dformat* is a text string containing special characters instructing that various fields be printed at various points in a manner similar to the formatting string used by printf. For example, the *dformat* " is ($0, $1 \)\n" would render the POINT 3, -4 which has a *format* of 32ii as " is (3, -4)\n".

Any character other than a $, ?, \, or a (in a *dformat* prints as itself. To print out one of $, ?, \, or (preceed it by a \. For example, to print out a $, use \$. Several special backslash sequences are provided as shortcuts. \n will cause a newline to be displayed while \t will cause a tab to be displayed. *o* where *o* is an octal number will display character number *o*.

A $ followed by a number *n* causes field number *n* to be displayed. The format of the displayed field depends on the formatting character used to describe it in the corresponding *format*. I.e., if a cardinal is described by 'c' it will print in decimal while if it is described by a 'x' it is displayed in hex.

If the field is not present in the property (this is possible with some properties), <field not available> is displayed instead. $*n*+ will display field number *n* then a comma then field number *n*+1 then another comma then ... until the last field defined. If field *n* is not defined, nothing is displayed. This is useful for a property that is a list of values.

A ? is used to start a conditional expression, a kind of if-then statement. ?*exp*(*text*) will display *text* if and only if *exp* evaluates to non-zero. This is useful for two things. First, it allows fields to be displayed if and only if a flag is set. And second, it allows a value such as a state number to be displayed as a name rather than as just a number. The syntax of *exp* is as follows:

exp	::= *term* | *term=exp* | !*exp*
term	::= *n* | $*n* | m*n*

The ! operator is a logical ''not'', changing 0 to 1 and any non-zero value to 0. = is an equality operator. Note that internally all expressions are evaluated as 32 bit numbers so -1 is not equal to 65535. = returns 1 if the two values are equal and 0 if not. *n* represents the constant value *n* while $*n* represents the value of field number *n*. m*n* is 1 if flag number *n* in the first field having format character 'm' in the corrsponding *format* is 1, 0 otherwise.

Examples: ?m3(count: $3\n) displays field 3 with a label of count if and only if flag number 3 (count starts at 0!) is on. ?$2=0(True)?!$2=0(False) displays the inverted value of field 2 as a boolean.

In order to display a property, *xprop* needs both a *format* and a *dformat*. Before *xprop* uses its default values of a *format* of 32x and a *dformat* of " = { $0+ }\n", it searches several places in an attempt to find more specific formats. First, a search is made using the name of the property. If this fails, a search is made using the type of the property. This allows type STRING to be defined with one set of formats while allowing property WM_NAME which is of type STRING to be defined with a different format. In this way, the display formats for a given type can be overridden for specific properties.

The locations searched are in order: the format if any specified with the property name (as in 8x WM_NAME), the formats defined by -f options in last to first order, the contents of the file specified by the -fs option if any, the contents of the file specified by the environmental variable XPROPFORMATS if any, and finally *xprop*'s built in file of formats.

The format of the files refered to by the -fs argument and the XPROPFORMATS variable is one or more lines of the following form:

name format [*dformat*]

Where *name* is either the name of a property or the name of a type, *format* is the *format* to be used with *name* and *dformat* is the *dformat* to be used with *name*. If *dformat* is not present, " = $0+\n" is assumed.

EXAMPLES

To display the name of the root window: *xprop* -root WM_NAME

To display the window manager hints for the clock: *xprop* -name xclock WM_HINTS

To display the start of the cut buffer: *xprop* -root -len 100 CUT_BUFFER0

To display the point size of the fixed font: *xprop* -font fixed POINT_SIZE

To display all the properties of window # 0x200007: *xprop* -id 0x200007

ENVIRONMENT

DISPLAY

To get default display.

XPROPFORMATS

Specifies the name of a file from which additional formats are to be obtained.

SEE ALSO

X(1), xwininfo(1)

COPYRIGHT

AUTHOR

Mark Lillibridge, MIT Project Athena

NAME

xrdb - X server resource database utility

SYNOPSIS

xrdb [-option ...] [*filename*]

DESCRIPTION

Xrdb is used to get or set the contents of the RESOURCE_MANAGER property on the root window of screen 0. You would normally run this program from your X startup file.

The resource manager (used by the Xlib routine *XGetDefault(3X)* and the X Toolkit) uses the RESOURCE_MANAGER property to get user preferences about color, fonts, and so on for applications. Having this information in the server (where it is available to all clients) instead of on disk, solves the problem in previous versions of X that required you to maintain *defaults* files on every machine that you might use. It also allows for dynamic changing of defaults without editting files.

For compatibility, if there is no RESOURCE_MANAGER property defined (either because xrdb was not run or if the property was removed), the resource manager will look for a file called *Xdefaults* in your home directory.

The *filename* (or the standard input if - or no input file is given) is optionally passed through the C preprocessor with the following symbols defined, based on the capabilities of the server being used:

BITS_PER_RGB=num
the number of significant bits in an RGB color specification. This is the log base 2 of the number of distinct shades of each primary that the hardware can generate. Note that it usually is not related to PLANES.

CLASS=visualclass
one of StaticGray, GrayScale, StaticColor, PseudoColor, TrueColor, DirectColor. This is the visual class of the root window of the default screen.

COLOR defined only if CLASS is one of StaticColor, PseudoColor, TrueColor, or DirectColor.

HEIGHT=num
the height of the default screen in pixels.

HOST=hostname
the hostname portion of the display to which you are connected.

PLANES=num
: the number of bit planes (the depth) of the root window of the default screen.

RELEASE=num
: the vendor release number for the server. The interpretation of this number will vary depending on VENDOR.

REVISION=num
: the X protocol minor version supported by this server (currently 0).

VERSION=num
: the X protocol major version supported by this server (should always be 11).

VENDOR=vendor
: a string specifying the vendor of the server.

WIDTH=num
: the width of the default screen in pixels.

X_RESOLUTION=num
: the x resolution of the default screen in pixels per meter.

Y_RESOLUTION=num
: the y resolution of the default screen in pixels per meter.

Lines that begin with an exclamation mark (!) are ignored and may be used as comments.

OPTIONS

xrdb program accepts the following options:

–help This option (or any unsupported option) will cause a brief description of the allowable options and parameters to be printed.

–display *display*
: This option specifies the X server to be used; see *X(1)*.

–n This option indicates that changes to the property (when used with *-load*) or to the resource file (when used with *-edit*) should be shown on the standard output, but should not be performed.

–quiet This option indicates that warning about duplicate entries should not be displayed.

-cpp *filename*
: This option specifies the pathname of the C preprocessor program to be used. Although *xrdb* was designed to use CPP, any program that acts as a filter and accepts the -D, -I, and -U options may be used.

-nocpp This option indicates that *xrdb* should not run the input file through a preprocessor before loading it into the RESOURCE_MANAGER property.

–symbols
This option indicates that the symbols that are defined for the preprocessor should be printed onto the standard output. It can be used in conjunction with **–query,** but not with the options that change the RESOURCE_MANAGER property.

–query This option indicates that the current contents of the RESOURCE_MANAGER property should be printed onto the standard output. Note that since preprocessor commands in the input resource file are part of the input file, not part of the property, they won't appear in the output from this option. The **–edit** option can be used to merge the contents of the property back into the input resource file without damaging preprocessor commands.

–load This option indicates that the input should be loaded as the new value of the RESOURCE_MANAGER property, replacing whatever was there (i.e. the old contents are removed). This is the default action.

–merge This option indicates that the input should be merged with, instead of replacing, the current contents of the RESOURCE_MANAGER property. Since *xrdb* can read the standard input, this option can be used to the change the contents of the RESOURCE_MANAGER property directly from a terminal or from a shell script.

–remove
This option indicates that the RESOURCE_MANAGER property should be removed from its window.

–retain This option indicates that the server should be instructed not to reset of *xrdb* is the first client.

–edit *filename*
This option indicates that the contents of the RESOURCE_MANAGER property should be edited into the given file, replacing any values already listed there. This allows you to put changes that you have made to your defaults back into your resource file, preserving any comments or preprocessor lines.

–backup *string*
: This option specifies a suffix to be appended to the filename used with **–edit** to generate a backup file.

–D*name[=value]*
: This option is passed through to the preprocessor and is used to define symbols for use with conditionals such as *#ifdef.*

–U*name* This option is passed through to the preprocessor and is used to remove any definitions of this symbol.

–I*directory*
: This option is passed through to the preprocessor and is used to specify a directory to search for files that are referenced with *#include.*

FILES

Generalizes *~/Xdefaults* files.

SEE ALSO

X(1), XGetDefault(3X), Xlib Resource Manager documentation

ENVIRONMENT

DISPLAY
: to figure out which display to use.

BUGS

The default for no arguments should be to query, not to overwrite, so that it is consistent with other programs.

COPYRIGHT

Copyright 1988, Digital Equipment Corporation.

AUTHORS

Phil Karlton, rewritten from the original by Jim Gettys

NAME

xrefresh - refresh all or part of an X screen

SYNOPSIS

xrefresh [-option ...]

DESCRIPTION

Xrefresh is a simple X program that causes all or part of your screen to be repainted. This is useful when system messages have messed up your screen. *Xrefresh* maps a window on top of the desired area of the screen and then immediately unmaps it, causing refresh events to be sent to all applications. By default, a window with no background is used, causing all applications to repaint ''smoothly.'' However, the various options can be used to indicate that a solid background (of any color) or the root window background should be used instead.

ARGUMENTS

–white Use a white background. The screen just appears to flash quickly, and then repaint.

–black Use a black background (in effect, turning off all of the electron guns to the tube). This can be somewhat disorienting as everything goes black for a moment.

–solid *color*
Use a solid background of the specified color. Try green.

–root Use the root window background.

–none This is the default. All of the windows simply repaint.

–geometry *WxH+X+Y*
Specifies the portion of the screen to be repainted; see *X(1)*.

–display *display*
This argument allows you to specify the server and screen to refresh; see *X(1)*.

X DEFAULTS

The *xrefresh* program uses the routine *XGetDefault(3X)* to read defaults, so its resource names are all capitalized.

Black, White, Solid, None, Root
Determines what sort of window background to use.

Geometry
Determines the area to refresh. Not very useful.

ENVIRONMENT

DISPLAY - To get default host and display number.

SEE ALSO

X(1)

BUGS

It should have just one default type for the background.

COPYRIGHT

Copyright 1988, Massachusetts Institute of Technology.
See *X(1)* for a full statement of rights and permissions.

AUTHORS

Jim Gettys, Digital Equipment Corp., MIT Project Athena

NAME

xscope - X Window Protocol Viewer

SYNOPSIS

xscope [option] ...

DESCRIPTION

Xscope sits in-between an X11 client and an X11 server and prints the contents of each request, reply, error, or event that is communicated between them. This information can be useful in debugging and performance tuning of X11 servers and clients.

To operate, *xscope* must know the host, port, and display to use to connect to the X11 server. In addition, it must know the port on which it should listen for X11 clients. Two cases are common:

(1) The X11 server is on the same host as *xscope*.
In this case, the input port for *xscope* should be selected as an X11 server on a different display, and the client DISPLAY argument adjusted to select *xscope* . For example, if the X11 server is on port 6000, display 1, then *xscope* can use port 6002 as its input port. The client can use display 1 for direct access to X11 or display 2 for access to *xscope*.

(2) The X11 server is on a different host than *xscope*.
In this case the same input and output ports can be used, and the host component of the DISPLAY is used to select *xscope* or X11.

ARGUMENTS

–i<input-port>
Specify the port that *xscope* will use to take requests from clients (defaults to 1). For X11, this port is automatically biased by 6000.

–o<output-port>
Determines the port that *xscope* will use to connect to X11 (defaults to 0). For X11, this port is automatically biased by 6000.

–h<host> Determines the host that *xscope* will use to find its X11 server.

–d<display>
Defines the display number. The display number is added to the input and output port to give the actual ports which are used by *xscope*.

–q Quiet output mode. Gives only the names of requests, replies, errors, and events, but does not indicate contents.

–v<print-level>

Determines the level of printing which *xscope* will provide. The print-level can be 0 (same as quiet mode), 1, 2, 3, 4. The larger numbers give more and more output. For example, a successful setup returns a string which is the name of the vendor of the X11 server. At level 1, the explicit field giving the length of the string is suppressed since it can be inferred from the string. At level 2 and above the length is explicitly printed.

EXAMPLES

```
xscope -v4 -hcleo -d0 -o0 -i1
```

This command would have xscope communicate with an X11 server on host ''cleo'', display 0; xscope itself would be available on the current host as display 1 (display of 0 plus the 1 of -i1). Verbose level 4.

```
xscope -q -d1 -o1 -o3
```

The X11 server for the current host, display 2 (1 for -d1 plus 1 for -o1) would be used by xscope which would run as display 4 (1 for -d1 plus 3 for -o3). Quite mode (verbose level 0).

SEE ALSO

X(1), X11 Protocol document (doc/Protocol/X11.protocol)

AUTHOR

James L. Peterson (MCC)

Copyright 1988, MCC

BUGS

Code has only been tested on Sun3's.

NAME

X - X Window System server

SYNOPSIS

X [:displaynumber] [-option ...] [ttyname]

DESCRIPTION

X is the generic name for the X Window System server. It is frequently a link or a copy of the appropriate server binary for driving the most frequently used server on a given machine. The sample server from MIT supports the following platforms:

Xqvss	Digital monochrome vaxstationII or II
Xqdss	Digital color vaxstationII or II
Xsun	Sun monochrome or color Sun 2, 3, or 4
Xhp	HP Topcat 9000s300
Xapollo	Apollo monochrome (Domain/IX 9.6)
Xibm	IBM APA and megapel PC/RT
XmacII	Apple monochrome Macintosh II
Xplx	Parallax color and video graphics controller

STARTING THE SERVER

The server is usually started from the X Display Manager program *xdm*. This utility is run from the system boot files and takes care of keeping the server running, prompting for usernames and passwords, and starting up the user sessions. It is easily configured for sites that wish to provide nice, consistent interfaces for novice users (loading convenient sets of resources, starting up a window manager, clock, and nice selection of terminal emulator windows).

Since *xdm* now handles automatic starting of the server in a portable way, the *-L* option to *xterm* is now considered obsolete. Support for starting a login window from 4.3bsd-derived */etc/ttys* files may not be included in future releases.

Installations that run more than one window system will still need to use the *xinit* utility. However, *xinit* is to be considered a tool for building startup scripts and is not intended for use by end users. Site adminstrators are **strongly** urged to build nicer interfaces for novice users.

When the sample server starts up, it takes over the display. If you are running on a workstation whose console is the display, you cannot log into the console while the server is running.

NETWORK CONNECTIONS

The sample server supports connections made using the following reliable byte-streams:

TCP/IP

The server listens on port htons(6000+*n*), where *n* is the display number.

Unix Domain

The sample server uses */tmp/X11-unix/X***n** as the filename for the socket, where *n* is the display number.

DECnet

The server responds to connections to object *X$X***n**, where *n* is the display number.

OPTIONS

All of the sample servers accept the following command line options:

–a *number*

sets pointer acceleration (i.e. the ratio of how much is reported to how much the user actually moved the pointer).

–auth *authorization-file*

Specifies a file which contains a collection of authorization records used to authenticate access.

bc disables certain kinds of error checking, for bug compatibility with previous releases (e.g., to work around bugs in R2 and R3 xterms and toolkits). Deprecated.

–bs disables backing store support on all screens.

–c turns off key-click.

c *volume* sets key-click volume (allowable range: 0-100).

-cc *class* sets the visual class for the root window of color screens. The class numbers are as specified in the X protocol. Not obeyed by all servers.

–co *filename*

sets name of RGB color database.

–f *volume*

sets feep (bell) volume (allowable range: 0-100).

–fc *cursorFont*

sets default cursor font.

–fn *font* sets the default font.

–fp *fontPath*

sets the search path for fonts.

–help prints a usage message.

–I causes all remaining command line arguments to be ignored.

–ld *kilobytes*
sets the data space limit of the server to the specified number of kilobytes. The default value is zero, making the data size as large as possible. A value of -1 leaves the data space limit unchanged. This option is not available in all operating systems.

–ls *kilobytes*
sets the stack space limit of the server to the specified number of kilobytes. The default value is zero, making the stack size as large as possible. A value of -1 leaves the stack space limit unchanged. This option is not available in all operating systems.

–logo turns on the X Window System logo display in the screen-saver. There is currently no way to change this from a client.

nologo turns off the X Window System logo display in the screen-saver. There is currently no way to change this from a client.

–p *minutes*
sets screen-saver pattern cycle time in minutes.

–r turns off auto-repeat.

r turns on auto-repeat.

–s *minutes*
sets screen-saver timeout time in minutes.

–su disables save under support on all screens.

–t *numbers*
sets pointer acceleration threshold in pixels (i.e. after how many pixels pointer acceleration should take effect).

–to *seconds*
sets default connection timeout in seconds.

tty*xx* ignored, for servers started the ancient way (from init).

v sets video-on screen-saver preference.

–v sets video-off screen-saver preference.

–wm forces the default backing-store of all windows to be WhenMapped; a cheap trick way of getting backing-store to apply to all windows.

–x *extension*
: loads the specified extension at init. Not supported in most implementations.

Many servers also have device-specific command line options. See the manual pages for the individual servers for more details.

SECURITY

The sample server implements a simplistic authorization protocol which uses data private to authorized clients and the server. The authorzation data is passed to the server in a private file named with the **-auth** command line option. If this file contains any authorization records, the local host is not automatically allowed access to the server, and only clients which send one of the authorization records contained in the file in the connection setup information will be allowed access. The authorization name supported is "MIT-MAGIC-COOKIE-1". See the *Xau* manual page for a description of the binary format of this file. Maintenence of this file, and distribution of its contents to remote sites for use there is left as an exercise for the reader.

The sample server also uses a host-based access control list for deciding whether or not to accept connections from clients on a particular machine. This list initially consists of the host on which the server is running as well as any machines listed in the file */etc/X***n***.hosts*, where **n** is the display number of the server. Each line of the file should contain either an Internet hostname (e.g. expo.lcs.mit.edu) or a DECnet hostname in double colon format (e.g. hydra::). There should be no leading or trailing spaces on any lines. For example:

```
joesworkstation
corporate.company.com
star::
bigcpu::
```

Users can add or remove hosts from this list and enable or disable access control using the *xhost* command from the same machine as the server. For example:

```
% xhost +janesworkstation
janesworkstation being added to access control list
% xhost -star::
public:: being removed from access control list
% xhost +
all hosts being allowed (access control disabled)
% xhost -
all hosts being restricted (access control enabled)
% xhost
```

```
access control enabled (only the following hosts are allowed)
joesworkstation
janesworkstation
corporate.company.com
bigcpu::
```

Unlike some window systems, X does not have any notion of window operation permissions or place any restrictions on what a client can do; if a program can connect to a display, it has full run of the screen. Sites that have better authentication and authorization systems (such as Kerberos) might wish to make use of the hooks in the libraries and the server to provide additional security models.

SIGNALS

The sample server attaches special meaning to the following signals:

SIGHUP This signal causes the server to close all existing connections, free all resources, and restore all defaults. It is sent by the display manager whenever the main user's main application (usually an *xterm* or window manager) exits to force the server to clean up and prepare for the next user.

SIGTERM
This signal causes the server to exit cleanly.

FONTS

Fonts are usually stored as individual files in directories. The list of directories in which the server looks when trying to open a font is controlled by the *font path.* Although most sites will choose to have the server start up with the appropriate font path (using the *-fp* option mentioned above), it can be overridden using the *xset* program.

The default font path for the sample server contains three directories:

/usr/lib/X11/fonts/misc
This directory contains several miscellaneous fonts that are useful on all systems. It contains a very small family of fixed-width fonts (**6x10**, **6x12**, **6x13**, **8x13**, **8x13bold**, and **9x15**) and the cursor font. It also has font name aliases for the commonly used fonts **fixed** and **variable.**

/usr/lib/X11/fonts/75dpi
This directory contains fonts contributed by Adobe Systems, Inc. and Digital Equipment Corporation and by Bitstream, Inc. for 75 dots per inch displays. An integrated selection of sizes, styles, and weights are provided for each family.

/usr/lib/X11/fonts/100dpi
: This directory contains versions of some of the fonts in the *75dpi* directory for 100 dots per inch displays.

Font databases are created by running the *mkfontdir* program in the directory containing the compiled versions of the fonts (the *.snf* files). Whenever fonts are added to a directory, *mkfontdir* should be rerun so that the server can find the new fonts. **If *mkfontdir* is not run, the server will not be able to find any fonts in the directory.**

DIAGNOSTICS

Too numerous to list them all. If run from *init(8)*, errors are logged in the file */usr/adm/X*msgs*,

FILES

/etc/X*.hosts	Initial access control list
/usr/lib/X11/fonts/misc, /usr/lib/X11/fonts/75dpi, /usr/lib/X11/fonts/100dpi	Font directories
/usr/lib/X11/rgb.txt	Color database
/tmp/.X11-unix/X*	Unix domain socket
/usr/adm/X*msgs	Error log file

SEE ALSO

X(1), xdm(1), mkfontdir(1), xinit(1), xterm(1), twm(1), xhost(1), xset(1), xsetroot(1), ttys(5), init(8), Xqdss(1), Xqvss(1), Xsun(1), Xapollo(1), XmacII(1) *X Window System Protocol, Definition of the Porting Layer for the X v11 Sample Server, Strategies for Porting the X v11 Sample Server, Godzilla's Guide to Porting the X V11 Sample Server*

BUGS

The option syntax is inconsistent with itself and *xset(1)*.

The acceleration option should take a numerator and a denominator like the protocol.

If *X* dies before its clients, new clients won't be able to connect until all existing connections have their TCP TIME_WAIT timers expire.

The color database is missing a large number of colors. However, there doesn't seem to be a better one available that can generate RGB values tailorable to particular displays.

The *xterm -L* method for starting an initial window from */etc/ttys* is completely inadequate and should be removed. People should use *xdm* instead.

COPYRIGHT

AUTHORS

The sample server was originally written by Susan Angebranndt, Raymond Drewry, Philip Karlton, and Todd Newman, with support from a cast of thouands. See also the file *doc/contributors* in the sample distribution for a more complete list.

NAME

xset - user preference utility for X

SYNOPSIS

xset [-display *display*] [-b] [b on/off] [b [*volume* [*pitch* [*duration*]]] [-c] [c on/off] [c [*volume*]] [[-+]fp[-+=] *path*[,*path*[,...]]] [fp default] [fp rehash] [[-]led [*integer*]] [led on/off] [m[ouse] [*acceleration* [*threshold*]]] [m[ouse] default] [p *pixel color*] [[-]r] [r on/off] [s [*length* [*period*]]] [s blank/noblank] [s expose/noexpose] [s on/off] [s default] [q]

DESCRIPTION

This program is used to set various user preference options of the display.

OPTIONS

−display *display*

This option specifies the server to use; see *X(1)*.

b

the **b** option controls bell volume, pitch and duration. This option accepts up to three numerical parameters, a preceding dash(-), or a 'on/off' flag. If no parameters are given, or the 'on' flag is used, the system defaults will be used. If the dash or 'off' are given, the bell will be turned off. If only one numerical parameter is given, the bell volume will be set to that value, as a percentage of its maximum. Likewise, the second numerical parameter specifies the bell pitch, in hertz, and the third numerical parameter specifies the duration in milliseconds. Note that not all hardware can vary the bell characteristics. The X server will set the characteristics of the bell as closely as it can to the user's specifications.

c

The **c** option controls key click. This option can take an optional value, a preceding dash(-), or an 'on/off' flag. If no parameter or the 'on' flag is given, the system defaults will be used. If the dash or 'off' flag is used, keyclick will be disabled. If a value from 0 to 100 is given, it is used to indicate volume, as a percentage of the maximum. The X server will set the volume to the nearest value that the hardware can support.

fp= *path,...*

The **fp=** sets the font path to the directories given in the path argument. The directories are interpreted by the server, not by the client, and are server-dependent. Directories that do not contain font databases created by *mkfontdir* will be ignored by the server.

fp *default*

The *default* argument causes the font path to be reset to the server's default.

fp *rehash*

The *rehash* argument causes the server to reread the font databases in the current font path. This is generally only used when adding new fonts to a font directory (after running *mkfontdir* to recreate the font database).

–fp or **fp–**

The **–fp** and **fp–** options remove elements from the current font path. They must be followed by a comma-separated list of directories.

+fp or **fp+**

This **+fp** and **fp+** options prepend and append elements to the current font path, respectively. They must be followed by a comma-separated list of directories.

led

The **led** option controls the keyboard LEDs. This controls the turning on or off of one or all of the LEDs. It accepts an optional integer, a preceding dash(-) or an 'on/off' flag. If no parameter or the 'on' flag is given, all LEDs are turned on. If a preceding dash or the flag 'off' is given, all LEDs are turned off. If a value between 1 and 32 is given, that LED will be turned on or off depending on the existence of a preceding dash. A common LED which can be controlled is the "Caps Lock" LED. "xset led 3" would turn led #3 on. "xset -led 3" would turn it off. The particular LED values may refer to different LEDs on different hardware.

m

The **m** option controls the mouse parameters. The parameters for the mouse are 'acceleration' and 'threshold'. The mouse, or whatever pointer the machine is connected to, will go 'acceleration' times as fast when it travels more than 'threshold' pixels in a short time. This way, the mouse can be used for precise alignment when it is moved slowly, yet it can be set to travel across the screen in a flick of the wrist when desired. One or both parameters for the **m** option can be omitted, but if only one is given, it will be interpreted as the acceleration. If no parameters or the flag 'default' is used, the system defaults will be set.

p

The **p** option controls pixel color values. The parameters are the color map entry number in decimal, and a color specification. The root background colors may be changed on some servers by altering the entries for BlackPixel and WhitePixel. Although these are often 0 and 1, they need not be. Also, a server may choose to allocate those colors privately, in which case an error will be generated. The map entry must not be a read-only color, or an error will result.

r The **r** option controls the autorepeat. If a preceding dash or the 'off' flag is used, autorepeat will be disabled. If no parameters or the 'on' flag is used, autorepeat will be enabled.

s The s option lets you set the screen saver parameters. This option accepts up to two numerical parameters, a 'blank/noblank' flag, an 'expose/noexpose' flag, an 'on/off' flag, or the 'default' flag. If no parameters or the 'default' flag is used, the system will be set to its default screen saver characteristics. The 'on/off' flags simply turn the screen saver functions on or off. The 'blank' flag sets the preference to blank the video (if the hardware can do so) rather than display a background pattern, while 'noblank' sets the preference to display a pattern rather than blank the video. The 'expose' flag sets the preference to allow window exposures (the server can freely discard window contents), while 'noexpose' sets the preference to disable screen saver unless the server can regenerate the screens without causing exposure events. The length and period parameters for the screen saver function determines how long the server must be inactive for screen saving to activate, and the period to change the background pattern to avoid burn in. The arguments are specified in seconds. If only one numerical parameter is given, it will be used for the length.

q The **q** option gives you information on the current settings.

These settings will be reset to default values when you log out.

Note that not all X implementations are guaranteed to honor all of these options.

SEE ALSO

X(1), Xserver(1), xmodmap(1), xrdb(1), xsetroot(1)

COPYRIGHT

AUTHOR

Bob Scheifler, MIT Laboratory for Computer Science
David Krikorian, MIT Project Athena (X11 version)

NAME

xsetroot – root window parameter setting utility for X

SYNOPSIS

xsetroot [-help] [-def] [-display *display*] [-cursor *cursorfile maskfile*] [-bitmap *filename*] [-mod *x y*] [-gray] [-grey] [-fg *color*] [-bg *color*] [-rv] [-solid *color*] [-name *string*]

DESCRIPTION

The *setroot* program allows you to tailor the appearance of the background ("root") window on a workstation display running X. Normally, you experiment with *xsetroot* until you find a personalized look that you like, then put the *xsetroot* command that produces it into your X startup file. If no options are specified, or if *-def* is specified, the window is reset to its default state. The *-def* option can be specified along with other options and only the non-specified characteristics will be reset to the default state.

Only one of the background color/tiling changing options (-solid, -gray, -grey, -bitmap, and -mod) may be specified at a time.

OPTIONS

The various options are as follows:

-help Print a usage message and exit.

-def Reset unspecified attributes to the default values. (Restores the background to the familiar gray mesh and the cursor to the hollow x shape.)

-cursor *cursorfile maskfile*
This lets you change the pointer cursor to whatever you want when the pointer cursor is outside of any window. Cursor and mask files are bitmaps (little pictures), and can be made with the *bitmap(1)* program. You probably want the mask file to be all black until you get used to the way masks work.

-bitmap *filename*
Use the bitmap specified in the file to set the window pattern. You can make your own bitmap files (little pictures) using the *bitmap(1)* program. The entire background will be made up of repeated "tiles" of the bitmap.

-mod *x y*
This is used if you want a plaid-like grid pattern on your screen. x and y are integers ranging from 1 to 16. Try the different combinations. Zero and negative numbers are taken as 1.

-gray Make the entire background gray. (Easier on the eyes.)

-grey Make the entire background grey.

-fg *color*
Use "color" as the foreground color. Foreground and background colors are meaningful only in combination with -cursor, -bitmap, or -mod.

-bg *color*
Use "color" as the background color.

-rv This exchanges the foreground and background colors. Normally the foreground color is black and the background color is white.

-solid *color*
This sets the background of the root window to the specified color. This option is only useful on color servers.

-name *string*
Set the name of the root window to "string". There is no default value. Usually a name is assigned to a window so that the window manager can use a text representation when the window is iconified. This option is unused since you can't iconify the background.

-display *display*
Specifies the server to connect to; see *X(1)*.

SEE ALSO
X(1), xset(1), xrdb(1)

COPYRIGHT

AUTHOR
Mark Lillibridge, MIT Project Athena

NAME

Xsgi - SGI Iris server for the X Window System

SYNOPSIS

Xsgi [:displaynumber] [-option ...]

DESCRIPTION

Xsgi is the name for the Silicon Graphics, Inc. X Window System server. The X Windows server provides an interface for programs written for the X Window System, Version 11, developed jointly by MIT's Project Athena and Digital Equipment Corporation, with contributions from many other companies. It was designed by Robert Scheifler, Jim Gettys, and others at MIT, and was based in part on the "W" windowing package developed by Paul Asente at Stanford University.

It is strongly recommended that you refer to the *X Window System User's Guide for Version 11*, by Tim O'Reilly, Valerie Quercia, and Linda Lamb (O'Reilly & Associates, ISBN 0-937175-29-3). Although this man page gives you all the information required to run X clients, the *X Window System User's Guide* contains additional information that is particularly useful to people who intend to develop their own clients.

In addition, you may find the following books to be useful: *Xlib Programming Manual (Volume I)*, by Adrian Nye, O'Reilly & Associates, ISBN 0-937175-26-9 *Xlib Reference Manual (Volume II)*, O'Reilly & Associates, ISBN 0-937175-27-7

STARTING THE SERVER

The X11 Window System runs on the same screen as the NeWS^{TM} window system, which allows GL, NeWS, and X11 clients to run simultaneously. However, there are some differences.

Xsgi is normally run by the *xstart* program during the NeWS login process. However, it may also be started by hand. For more information on starting *X* from NeWS, please refer to the *xstart(1)* man page.

OPTIONS

Xsgi accepts the following command line options:

–a *number-of-pixels*
Set the mouse acceleration threshold. Currently not supported.

–t *number-of-pixels*
Set the mouse threshold. Currently not supported.

–auth string
Select authorization file.

bc Enable backwards compatibility mode. Currently not supported.

–bs disables backing store support on all screens.

–c Turns off key-click. Currently not supported.

c *percent*
Key-click volume (0-100). Currently not supported.

–r Turns off auto-repeat. **r** Turns on auto-repeat.

–cc *visual-number*
Set the default color visual class.

–f *percent*
Set the bell base volume.

–logo turns on the X Window System logo display in the screen-saver. There is currently no way to change this from a client.

nologo turns off the X Window System logo display in the screen-saver. There is currently no way to change this from a client.

–p *minutes*
sets screen-saver pattern cycle time in minutes. The default is 10 minutes.

–s *minutes*
sets screen-saver timeout time in minutes. The screen saver may be disabled by setting the timeout to 0. This is the default.

–to *seconds*
sets default screensaver timeout in seconds. The screen saver may be disabled by setting the timeout to 0. This is the default.

v Turn on video blanking for screen-saver. Currently not supported.

–v Turn off video blanking for screen-saver. Currently not supported.

–su disables save under support on all screens.

–co *filename*
sets name of RGB color database. The default is */usr/lib/X11/rbg.txt*

–help prints a usage message

–fp *fontPath*
sets the search path for fonts. This defaults to */usr/lib/X11/fonts/misc/*, */usr/lib/X11/fonts/100dpi/*, */usr/lib/X11/fonts/75dpi/*, */usr/lib/fmfonts/*

–fc *cursorFont*
: sets default cursor font. This defaults to *cursor.*

–fn *font* sets the default font. This defaults to *fixed.*

–x *extension-name*
: Loads the named extension at init time.

–wm forces the default backing-store of all windows to be WhenMapped; a not very good way of getting backing-store to apply to all windows.

–static Set the default visual to an 8-bit StaticColor visual. All 256 entries in the default colormap are preallocated.

–gl Sets the default visual to an 8-bit PseudoColor visual with the colors 0-15 and 32-63 in the default colormap preallocated for use by the gl. This leaves 208 writable color cells for X.

–envm Sets the default visual to an 8-bit PseudoColor visual with all colors in the default colormap except 16-31 preallocated. GL and NeWS clients consider these colors off limits.

–pseudo The default visual is set to an 8-bit PseudoColor visual with no colors in the default colormap preallocated. There is no color map arbitration provided. GL and NeWS clients may also overwrite the same colors.

SECURITY

Xsgi uses an access control list for deciding whether or not to accept connections from clients on a particular machine. This list initially consists of the host on which the server is running as well as any machines listed in the file */etc/X***n***.hosts*, where **n** is the display number of the server. Each line of the file should contain an Internet hostname (e.g. expo.lcs.mit.edu).

Users can add or remove hosts from this list and enable or disable access control using the *xhost* command from the same machine as the server. Please refer to the *xhost(1)* man page for more information.

Unlike some window systems, X does not have any notion of window operation permissions or place any restrictions on what a client can do; if a program can connect to a display, it has full run of the screen. See also *xauth(1).*

SIGNALS

The sample server attaches special meaning to the following signals:

SIGHUP This signal causes *Xsgi* to close all existing connections, free all resources, and restore all defaults. It is sent by the display manager whenever the main user's main application (usually an *xterm* or window manager) exits to force the server to clean up and prepare for the next user.

SIGTERM

This signal causes the *Xsgi* to exit cleanly.

SEE ALSO

X(1), Xserver(1), xdm(1), mkfontdir(1), xinit(1), xterm(1), twm(1), xhost(1), xset(1), xsetroot(1), xstart(1), xauth(1)

BUGS

The option syntax is inconsistent with itself and *xset(1)*.

If *X* dies before its clients, new clients won't be able to connect until all existing connections have their TCP TIME_WAIT timers expire.

The -a, bc, -c, c, -t, ttyxx, v, and -v options are not supported.

Backing store does not work correctly unless an *X* window manager is running. If an *X* window manager (eg. *twm*, *mwm*) is not used, then the -bs option should be used.

Xsgi ignores the CAPSLOCK key.

ADDITIONAL DOCUMENTATION

Additional documentation for developers of X Window System clients is available on-line and directly from MIT. To obtain materials directly from MIT, contact:

MIT Software Distribution Center
Technology Licensing Office
MIT E32-300
77 Massachusetts Avenue
Cambridge, MA 02139

Their telephone number is (617) 253-6966, and the "X Ordering Hotline" is (617) 258-8330.

COPYRIGHT

Copyright 1989, 1990, Silicon Graphics Inc.
Copyright 1987, 1988, Massachusetts Institute of Technology.
See *X(1)* for a full statement of rights and permissions.

NAME

xshowcmap – show colormap

SYNOPSIS

xshowcmap [**options**]

DESCRIPTION

Xshowcmap displays the contents of the currently active colormap in a window. The created window shows a square for every color currently defined in the servers active colormap. The number of squares is the number of colormap-cells the server supports.

Xshowcmap has been specially written to aid server debugging/verification.

To leave *xshowcmap* type 'q' while the cursor is in its window.

The following options are valid:

-bd color
as usual - change border color

-bw number
as usual - change border width

host:display
Name of the display.

BUGS

There might be some - it has been tested with Siemens servers running both monochrome and color, and with uVax Color-Workstations running color.

AUTHORS

Claus Gittinger (..!decvax!unido!sinix!claus)

NAME

X Standards

SYNOPSIS

The major goal of the MIT X Consortium is to promote cooperation within the computer industry in the creation of standard software interfaces at all layers in the X Window System environment. The status of various standards and proposed standards, and of the software in the X11R4 distribution, is explained below.

STANDARDS

The following documents are MIT X Consortium standards:

X Window System Protocol
X Version 11, Release 4
Robert W. Scheifler

Xlib - C Language X Interface
X Version 11, Release 4
James Gettys, Robert W. Scheifler, Ron Newman

X Toolkit Intrinsics - C Language Interface
X Version 11, Release 4
Joel McCormack, Paul Asente, Ralph R. Swick

Bitmap Distribution Format
Version 2.1

Inter-Client Communication Conventions Manual
Version 1.0
David S. H. Rosenthal

Compound Text Encoding
Version 1.1
Robert W. Scheifler

X Logical Font Description Conventions
Version 1.3
Jim Flowers

X Display Manager Control Protocol
Version 1.0
Keith Packard

X11 Nonrectangular Window Shape Extension

Version 1.0
Keith Packard

DRAFT STANDARDS

The following documents are draft standards of the MIT X Consortium. To become standards, further ''proof of concept'' is required, in the form of working implementations. The specifications may be subject to incompatible changes if implementation efforts uncover significant problems.

PEX Protocol Specification
Version 4.0P
Randi J. Rost (editor)

Extending X for Double-Buffering, Multi-Buffering, and Stereo
Version 3.2
Jeffrey Friedberg, Larry Seiler, Jeff Vroom

PUBLIC REVIEW DRAFTS

The following documents are out for Public Review for adoption as MIT X Consortium standards.

X11 Input Extension Protocol Specification
Public Review Draft
George Sachs, Mark Patrick

X11 Input Extension Library Specification
Public Review Draft
Mark Patrick, George Sachs

INCLUDE FILES

The following include files are part of the Xlib standard. The C++ support in these header files is experimental, and is not yet part of the standard.

<X11/X.h>
<X11/Xatom.h>
<X11/Xproto.h>
<X11/Xprotostr.h>
<X11/keysym.h>
<X11/keysymdef.h>
<X11/Xlib.h>
<X11/Xresource.h>
<X11/Xutil.h>
<X11/cursorfont.h>

<X11/X10.h>
<X11/Xlibint.h>

The following include files are part of the X Toolkit Intrinsics standard. The C++ support in these header files is experimental, and is not yet part of the standard.

<X11/Composite.h>
<X11/CompositeP.h>
<X11/ConstrainP.h>
<X11/Constraint.h>
<X11/Core.h>
<X11/CoreP.h>
<X11/Intrinsic.h>
<X11/IntrinsicP.h>
<X11/Object.h>
<X11/ObjectP.h>
<X11/Quarks.h>
<X11/RectObj.h>
<X11/RectObjP.h>
<X11/Shell.h>
<X11/ShellP.h>
<X11/StringDefs.h>
<X11/Vendor.h>
<X11/VendorP.h>

The following include file is part of the Nonrectangular Window Shape Extension standard.

<X11/extensions/shape.h>

The following include file is part of the Multi-Buffering draft standard.

<X11/extensions/multibuf.h>

NON STANDARDS

The X11R4 distribution contains *sample* implementations, not *reference* implementations. Although much of the code is believed to be correct, the code should be assumed to be in error wherever it conflicts with the specification.

At the public release of X11R4, the only MIT X Consortium standards are the ones listed above. No other documents, include files, or software in X11R4 carry special status within the X Consortium. For example, none of the following are standards: internal interfaces of the sample server; the MIT-SHM extension; the Input Synthesis extension; the Athena Widget Set;

the Xmu library; the Xau library; CLX, the Common Lisp interface to X (although a Consortium review is finally expected to begin); the RGB database; the fonts distributed with X11R4; the applications distributed with X11R4; the include files <X11/XWDFile.h> and <X11/Xos.h>; the bitmap files in <X11/bitmaps>.

NAME

xstart – start up the sgi X server as a NeWS client

SYNOPSIS

xstart [-c] [*X command*]

DESCRIPTION

xstart will start the sgi X server in background. It is intended to be executed from *init.ps,* the PostScript file which gets executed during the bringup of *SGI's NeWS*-based window system. However, *xstart* usage is not confined to init.ps; it can be executed at any time so long as *NeWS* is running.

If *xstart* is given the **–c** option, it will conditionally start up X, depending on whether *xSGINeWS* is configured. The command

chkconfig xSGINeWS on

will configure *xSGINeWS* on a per-machine basis. However, if the user has a file *$HOME/.xSGINeWS* containing either "on" or "off," that file will override the system-wide config option.

The actual command which *xstart* will use for starting X is determined by (in order of decreasing priority):

1. If there are arguments to *xstart* other than **-c**, those arguments specify the command.

2. If there are no arguments to *xstart* (except for possibly **-c**), the command is taken from the ascii file *$HOME/.xSGINeWS.cmd.*

3. If *$HOME/.xSGINeWS.cmd* doesn't exist, the command is taken from the ascii file */etc/gl/xSGINeWS.cmd.*

4. If */etc/gl/.xSGINeWS.cmd* doesn't exist, the default command used by *xstart* is

/usr/bin/X11/Xsgi –bs –envm.

Before starting X, *xstart* changes the current working directory to /tmp. Hence care should be taken if the user supplies an X server command (via *xstart* arguments or one of the *.cmd* files), and if the X server name does not include a full pathname.

A side effect of starting X by *xstart* is that X server will log its error messages to */usr/adm/X0msgs,* provided that file is writable.

WARNING

The purpose of *xstart* is simply to conditionally start up *Xsgi* from *init.ps.* At some point in the future, *xstart* will not be applicable and will no longer be supported.

NAME

xstdcmap - X standard colormap utility

SYNOPSIS

xstdcmap [-all] [-best] [-blue] [-default] [-delete *map*] [-display *display*] [-gray] [-green] [-help] [-red] [-verbose]

DESCRIPTION

The *xstdcmap* utility can be used to selectively define standard colormap properties. It is intended to be run from a user's X startup script to create standard colormap definitions in order to facilitate sharing of scarce colormap resources among clients. Where at all possible, colormaps are created with read-only allocations.

OPTIONS

The following options may be used with *xstdcmap*:

–all This option indicates that all six standard colormap properties should be defined on each screen of the display. Not all screens will support visuals under which all six standard colormap properties are meaningful. *xstdcmap* will determine the best allocations and visuals for the colormap properties of a screen. Any previously existing standard colormap properties will be replaced.

–best This option indicates that the RGB_BEST_MAP should be defined.

–blue This option indicates that the RGB_BLUE_MAP should be defined.

–default This option indicates that the RGB_DEFAULT_MAP should be defined.

–delete *map*
This option specifies that a standard colormap property should be removed. *mapP may be one of: default, best, red, green, blue, or gray.*

–display *display*
This option specifies the host and display to use; see *X(1)*.

–gray This option indicates that the RGB_GRAY_MAP should be defined.

–green This option indicates that the RGB_GREEN_MAP should be defined.

–help This option indicates that a brief description of the command line arguments should be printed on the standard error. This will be done whenever an unhandled argument is given to *xstdcmap*.

−red This option indicates that the RGB_RED_MAP should be defined.

−verbose
This option indicates that *xstdcmap* should print logging information as it parses its input and defines the standard colormap properties.

ENVIRONMENT

DISPLAY
to get default host and display number.

SEE ALSO

X(1)

COPYRIGHT

AUTHOR

Donna Converse, MIT X Consortium

NAME

xstr – extract strings from C programs to implement shared strings

SYNOPSIS

xstr [–v] [[**–c**] [–] [file]

DESCRIPTION

Xstr maintains a file *strings* into which strings in component parts of a large program are hashed. These strings are replaced with references to this common area. This serves to implement shared constant strings, most useful if they are also read-only. The –v flag makes *xstr* verbose.

The command

```
xstr –c name
```

will extract the strings from the C source in name, replacing string references by expressions of the form (&xstr[number]) for some number. An appropriate declaration of *xstr* is prepended to the file. The resulting C text is placed in the file *x.c*, to then be compiled. The strings from this file are placed in the *strings* data base if they are not there already. Repeated strings and strings which are suffices of existing strings do not cause changes to the data base.

After all components of a large program have been compiled a file *xs.c* declaring the common *xstr* space can be created by a command of the form

```
xstr
```

This *xs.c* file should then be compiled and loaded with the rest of the program. If possible, the array can be made read-only (shared) saving space and swap overhead.

Xstr can also be used on a single file. A command

```
xstr name
```

creates files *x.c* and *xs.c* as before, without using or affecting any *strings* file in the same directory.

It may be useful to run *xstr* after the C preprocessor if any macro definitions yield strings or if there is conditional code which contains strings which may not, in fact, be needed. *Xstr* reads from its standard input when the argument '–' is given. An appropriate command sequence for running *xstr* after the C preprocessor is:

```
cc –E  name.c | xstr –c –
cc –c x.c
mv x.o name.o
```

Xstr does not touch the file *strings* unless new items are added, thus *make* can avoid remaking *xs.o* unless truly necessary.

FILES

strings	Data base of strings
x.c	Massaged C source
xs.c	C source for definition of array 'xstr'
/tmp/xs*	Temp file when 'xstr name' doesn't touch *strings*

SEE ALSO

mkstr(1)

AUTHOR

William Joy

BUGS

If a string is a suffix of another string in the data base, but the shorter string is seen first by *xstr* both strings will be placed in the data base, when just placing the longer one there will do.

NAME

xterm – terminal emulator for X

SYNOPSIS

xterm [-*toolkitoption* ...] [-option ...]

DESCRIPTION

The *xterm* program is a terminal emulator for the X Window System. It provides DEC VT102 and Tektronix 4014 compatible terminals for programs that can't use the window system directly. If the underlying operating system supports terminal resizing capabilities (for example, the SIGWINCH signal in systems derived from 4.3bsd), *xterm* will use the facilities to notify programs running in the window whenever it is resized.

The VT102 and Tektronix 4014 terminals each have their own window so that you can edit text in one and look at graphics in the other at the same time. To maintain the correct aspect ratio (height/width), Tektronix graphics will be restricted to the largest box with a 4014's aspect ratio that will fit in the window. This box is located in the upper left area of the window.

Although both windows may be displayed at the same time, one of them is considered the ''active'' window for receiving keyboard input and terminal output. This is the window that contains the text cursor and whose border highlights whenever the pointer is in either window. The active window can be chosen through escape sequences, the ''Modes'' menu in the VT102 window, and the ''Tektronix'' menu in the 4014 window.

OPTIONS

The *xterm* terminal emulator accepts all of the standard X Toolkit command line options as well as the following (if the option begins with a '+' instead of a '–', the option is restored to its default value):

–help This causes *xterm* to print out a verbose message describing its options.

–132 Normally, the VT102 DECCOLM escape sequence that switches between 80 and 132 column mode is ignored. This option causes the DECCOLM escape sequence to be recognized, and the *xterm* window will resize appropriately.

–ah This option indicates that *xterm* should always highlight the text cursor and borders. By default, *xterm* will display a hollow text cursor whenever the focus is lost or the pointer leaves the window.

+ah This option indicates that *xterm* should do text cursor highlighting.

−b *number*
: This option specifies the size of the inner border (the distance between the outer edge of the characters and the window border) in pixels. The default is 2.

−cc *characterclassrange*:*value*[,...]
: This sets classes indicated by the given ranges for using in selecting by words. See the section specifying character classes.

−cn This option indicates that newlines should not be cut in line-mode selections.

+cn This option indicates that newlines should be cut in line-mode selections.

−cr *color*
: This option specifies the color to use for text cursor. The default is to use the same foreground color that is used for text.

−cu This option indicates that *xterm* should work around a bug in the *curses*(3x) cursor motion package that causes the *more*(1) program to display lines that are exactly the width of the window and are followed by a line beginning with a tab to be displayed incorrectly (the leading tabs are not displayed).

+cu This option indicates that that *xterm* should not work around the *curses*(3x) bug mentioned above.

−e *program [arguments ...]*
: This option specifies the program (and its command line arguments) to be run in the *xterm* window. It also sets the window title and icon name to be the basename of the program being executed if neither *-T* nor *-n* are given on the command line. **This must be the last option on the command line.**

−fb *font* This option specifies a font to be used when displaying bold text. This font must be the same height and width as the normal font. If only one of the normal or bold fonts is specified, it will be used as the normal font and the bold font will be produced by overstriking this font. The default is to do overstriking of the normal font.

−j This option indicates that *xterm* should do jump scrolling. Normally, text is scrolled one line at a time; this option allows *xterm* to move multiple lines at a time so that it doesn't fall as far behind. Its use is strongly recommended since it make *xterm* much faster when scanning through large amounts of text. The VT100 escape sequences for enabling and disabling smooth scroll as well as the ''Modes'' menu can be used to turn this feature on or off.

+j This option indicates that *xterm* should not do jump scrolling.

–l This option indicates that *xterm* should send all terminal output to a log file as well as to the screen. This option can be enabled or disabled using the ''xterm X11'' menu.

+l This option indicates that *xterm* should not do logging.

–lf *filename*
This option specifies the name of the file to which the output log described above is written. If *file* begins with a pipe symbol (|), the rest of the string is assumed to be a command to be used as the endpoint of a pipe. The default filename is ''**XtermLog.***XXXXX*'' (where *XXXXX* is the process id of *xterm*) and is created in the directory from which *xterm* was started (or the user's home directory in the case of a login window).

–ls This option indicates that the shell that is started in the *xterm* window be a login shell (i.e. the first character of argv[0] will be a dash, indicating to the shell that it should read the user's .login or .profile).

+ls This option indicates that the shell that is started should not be a login shell (i.e. it will be a normal ''subshell'').

–mb This option indicates that *xterm* should ring a margin bell when the user types near the right end of a line. This option can be turned on and off from the ''Modes'' menu.

+mb This option indicates that margin bell should not be rung.

–mc milliseconds
This option specifies the maximum time between multi-click selections.

–ms *color*
This option specifies the color to be used for the pointer cursor. The default is to use the foreground color.

–nb *number*
This option specifies the number of characters from the right end of a line at which the margin bell, if enabled, will ring. The default is 10.

–rw This option indicates that reverse-wraparound should be allowed. This allows the cursor to back up from the leftmost column of one line to the rightmost column of the previous line. This is very useful for editing long shell command lines and is encouraged. This option can be turned on and off from the ''Modes'' menu.

+rw This option indicates that reverse-wraparound should not be allowed.

−s This option indicates that *xterm* may scroll asynchronously, meaning that the screen does not have to be kept completely up to date while scrolling. This allows *xterm* to run faster when network latencies are very high and is typically useful when running across a very large internet or many gateways.

+s This option indicates that *xterm* should scroll synchronously.

−sb This option indicates that some number of lines that are scrolled off the top of the window should be saved and that a scrollbar should be displayed so that those lines can be viewed. This option may be turned on and off from the ''Modes'' menu.

+sb This option indicates that a scrollbar should not be displayed.

−sf This option indicates that Sun Function Key escape codes should be generated for function keys.

+sf This option indicates that the standard escape codes should be generated for function keys.

−si This option indicates that output to a window should not automatically reposition the screen to the bottom of the scrolling region. This option can be turned on and off from the ''Modes'' menu.

+si This option indicates that output to a window should cause it to scroll to the bottom.

−sk This option indicates that pressing a key while using the scrollbar to review previous lines of text should cause the window to be repositioned automatically in the normal position at the bottom of the scroll region.

+sk This option indicates that pressing a key while using the scrollbar should not cause the window to be repositioned.

−sl *number*
This option specifies the number of lines to save that have been scrolled off the top of the screen. The default is 64.

−t This option indicates that *xterm* should start in Tektronix mode, rather than in VT102 mode. Switching between the two windows is done using the ''Modes'' menus.

+t This option indicates that *xterm* should start in VT102 mode.

−tm *string*
: This option specifies a series of terminal setting keywords followed by the characters that should be bound to those functions, similar to the *stty* program. Allowable keywords include: intr, quit, erase, kill, eof, eol, swtch, start, stop, brk, susp, dsusp, rprnt, flush, weras, and lnext. Control characters may be specified as ^char (e.g. ^c or ^u) and ^? may be used to indicate delete.

−tn *name*
: This option specifies the name of the terminal type to be set in the TERM environment variable. This terminal type must exist in the *termcap(5)* database and should have *li#* and *co#* entries.

−ut This option indicates that *xterm* shouldn't write a record into the the system log file */etc/utmp.*

+ut This option indicates that *xterm* should write a record into the system log file */etc/utmp.*

−vb This option indicates that a visual bell is preferred over an audible one. Instead of ringing the terminal bell whenever a Control-G is received, the window will be flashed.

+vb This option indicates that a visual bell should not be used.

−wf This option indicates that *xterm* should wait for the window to be mapped the first time before starting the subprocess so that the initial terminal size settings and environment variables are correct. It the application's responsibility to catch subsequent terminal size changes.

+wf This option indicates that *xterm* show not wait before starting the subprocess.

−C This option indicates that this window should receive console output. This is not supported on all systems.

−S*ccn* This option specifies the last two letters of the name of a pseudoterminal to use in slave mode, plus the number of the inherited file descriptor. The option is parsed "%c%c%d". This allows *xterm* to be used as an input and output channel for an existing program and is sometimes used in specialized applications.

The following command line arguments are provided for compatibility with older versions. They may not be supported in the next release as the X Toolkit provides standard options that accomplish the same task.

%geom This option specifies the preferred size and position of the Tektronix window. It is shorthand for specifying the "**tekGeometry*" resource.

#geom This option specifies the preferred position of the icon window. It is shorthand for specifying the "**iconGeometry*" resource.

–T *string*
This option specifies the title for *xterm*'s windows. It is equivalent to **-title.**

–n *string*
This option specifies the icon name for *xterm*'s windows. It is shorthand for specifying the "**iconName*" resource. Note that this is not the same as the toolkit option **-name** (see below). The default icon name is the application name.

–r This option indicates that reverse video should be simulated by swapping the foreground and background colors. It is equivalent to **-reversevideo** or **-rv.**

–w *number*
This option specifies the width in pixels of the border surrounding the window. It is equivalent to **-borderwidth** or **-bw.**

The following standard X Toolkit command line arguments are commonly used with *xterm*:

–bg *color*
This option specifies the color to use for the background of the window. The default is "white."

–bd *color*
This option specifies the color to use for the border of the window. The default is "black."

–bw *number*
This option specifies the width in pixels of the border surrounding the window.

–fg *color*
This option specifies the color to use for displaying text. The default is "black".

–fn *font* This option specifies the font to be used for displaying normal text. The default is *fixed*.

–name *name*
This option specifies the application name under which resources are to be obtained, rather than the default executable file name. *Name* should not contain "." or "*" characters.

–title *string*
: This option specifies the window title string, which may be displayed by window managers if the user so chooses. The default title is the command line specified after the **-e** option, if any, otherwise the application name.

–rv This option indicates that reverse video should be simulated by swapping the foreground and background colors.

–geometry *geometry*
: This option specifies the preferred size and position of the VT102 window; see *X(1)*.

–display *display*
: This option specifies the X server to contact; see *X(1)*.

–xrm *resourcestring*
: This option specifies a resource string to be used. This is especially useful for setting resources that do not have separate command line options.

–iconic This option indicates that *xterm* should ask the window manager to start it as an icon rather than as the normal window.

RESOURCES

The program understands all of the core X Toolkit resource names and classes as well as:

iconGeometry (class **IconGeometry**)
: Specifies the preferred size and position of the application when iconified. It is not necessarily obeyed by all window managers.

termName (class **TermName**)
: Specifies the terminal type name to be set in the TERM environment variable.

title (class **Title**)
: Specifies a string that may be used by the window manager when displaying this application.

ttyModes (class **TtyModes**)
: Specifies a string containing terminal setting keywords and the characters to which they may be bound. Allowable keywords include: intr, quit, erase, kill, eof, eol, swtch, start, stop, brk, susp, dsusp, rprnt, flush, weras, and lnext. Control characters may be specified as ^char (e.g. ^c or ^u) and ^? may be used to indicate delete. This is very useful for overriding the default terminal settings without having to do an *stty* every time an *xterm* is started.

utmpInhibit (class **UtmpInhibit**)
: Specifies whether or not *xterm* should try to record the user's terminal in */etc/utmp.*

sunFunctionKeys (class **SunFunctionKeys**)
: Specifies whether or not Sun Function Key escape codes should be generated for function keys instead of standard escape sequences.

The following resources are specified as part of the *vt100* widget (class *VT100*):

allowSendEvents (class **AllowSendEvents**)
: Specifies whether or not synthetic key and button events (generated using the X protocol SendEvent request) should be interpreted or discarded. The default is ''false'' meaning they are discarded. Note that allowing such events creates a very large security hole.

alwaysHighlight (class **AlwaysHighlight**)
: Specifies whether or not *xterm* should always display a highlighted text cursor. By default, a hollow text cursor is displayed whenever the pointer moves out of the window or the window loses the input focus.

boldFont (class **Font**)
: Specifies the name of the bold font to use instead of overstriking.

c132 (class **C132**)
: Specifies whether or not the VT102 DECCOLM escape sequence should be honored. The default is ''false.''

charClass (class **CharClass**)
: Specifies comma-separated lists of character class bindings of the form [*low-*]*high:value.* These are used in determining which sets of characters should be treated the same when doing cut and paste. See the section on specifying character classes.

curses (class **Curses**)
: Specifies whether or not the last column bug in *curses*(3x) should be worked around. The default is ''false.''

background (class **Background**)
: Specifies the color to use for the background of the window. The default is ''white.''

foreground (class **Foreground**)
Specifies the color to use for displaying text in the window. Setting the class name instead of the instance name is an easy way to have everything that would normally appear in the "text" color change color. The default is "black."

cursorColor (class **Foreground**)
Specifies the color to use for the text cursor. The default is "black."

eightBitInput (class **EightBitInput**)
Specifies whether or not eight-bit characters should be accepted. The default is "true."

font (class **Font**)
Specifies the name of the normal font. The default is "vtsingle."

font1 (class **Font1**)
Specifies the name of the first alternate font.

font2 (class **Font2**)
Specifies the name of the second alternate font.

font3 (class **Font3**)
Specifies the name of the third alternate font.

font4 (class **Font4**)
Specifies the name of the fourth alternate font.

geometry (class **Geometry**)
Specifies the preferred size and position of the VT102 window.

internalBorder (class **BorderWidth**)
Specifies the number of pixels between the characters and the window border. The default is 2.

jumpScroll (class **JumpScroll**)
Specifies whether or not jump scroll should be used. The default is "true".

logFile (class **Logfile**)
Specifies the name of the file to which a terminal session is logged. The default is "**XtermLog.***XXXXX*" (where *XXXXX* is the process id of *xterm*).

logging (class **Logging**)
Specifies whether or not a terminal session should be logged. The default is "false."

logInhibit (class **LogInhibit**)
Specifies whether or not terminal session logging should be inhibited. The default is "false."

loginShell (class **LoginShell**)
Specifies whether or not the shell to be run in the window should be started as a login shell. The default is "false."

marginBell (class **MarginBell**)
Specifies whether or not the bell should be run when the user types near the right margin. The default is "false."

multiScroll (class **MultiScroll**)
Specifies whether or not asynchronous scrolling is allowed. The default is "false."

multiClickTime (class **MultiClickTime**)
Specifies the maximum time in milliseconds between multi-clock select events. The default is 250 milliseconds.

multiScroll (class **MultiScroll**)
Specifies whether or not scrolling should be done asynchronously. The default is "false."

nMarginBell (class **Column**)
Specifies the number of characters from the right margin at which the margin bell should be run, when enabled.

pointerColor (class **Foreground**)
Specifies the foreground color of the pointer. The default is "XtDefaultForeground."

pointerColorBackground (class **Background**)
Specifies the background color of the pointer. The default is "XtDefaultBackground."

pointerShape (class **Cursor**)
Specifies the name of the shape of the pointer. The default is "xterm."

reverseVideo (class **ReverseVideo**)
Specifies whether or not reverse video should be simulated. The default is "false."

reverseWrap (class **ReverseWrap**)
Specifies whether or not reverse-wraparound should be enabled. The default is "false."

saveLines (class **SaveLines**)
Specifies the number of lines to save beyond the top of the screen when a scrollbar is turned on. The default is 64.

scrollBar (class **ScrollBar**)
Specifies whether or not the scrollbar should be displayed. The default is "false."

scrollInput (class **ScrollCond**)
Specifies whether or not output to the terminal should automatically cause the scrollbar to go to the bottom of the scrolling region. The default is "true."

scrollKey (class **ScrollCond**)
Specifies whether or not pressing a key should automatically cause the scrollbar to go to the bottom of the scrolling region. The default is "false."

scrollLines (class **ScrollLines**)
Specifies the number of lines that the *scroll-back* and *scroll-forw* actions should use as a default. The default value is 1.

signalInhibit (class **SignalInhibit**)
Specifies whether or not the entries in the "xterm X11" menu for sending signals to *xterm* should be disallowed. The default is "false."

tekGeometry (class **Geometry**)
Specifies the preferred size and position of the Tektronix window.

tekInhibit (class **TekInhibit**)
Specifies whether or not Tektronix mode should be disallowed. The default is "false."

tekSmall (class **TekSmall**)
Specifies whether or not the Tektronix mode window should start in its smallest size if no explicit geometry is given. This is useful when running *xterm* on displays with small screens. The default is "false."

tekStartup (class **TekStartup**)
Specifies whether or not *xterm* should start up in Tektronix mode. The default is "false."

titeInhibit (class **TiteInhibit**)
Specifies whether or not *xterm* should remove remove *ti* or *te* termcap entries (used to switch between alternate screens on startup of many screen-oriented programs) from the TERMCAP string.

translations (class **Translations**)
Specifies the key and button bindings for menus, selections, ''programmed strings'', etc. See **ACTIONS** below.

visualBell (class **VisualBell**)
Specifies whether or not a visible bell (i.e. flashing) should be used instead of an audible bell when Control-G is received. The default is ''false.''

waitForMap (class **WaitForMap**)
Specifies whether or not *xterm* should wait for the initial window map before starting the subprocess. The default is ''false.''

The following resources are specified as part of the *tek4014* widget (class *Tek4014*):

width (class **Width**)
Specifies the width of the Tektronix window in pixels.

height (class **Height**)
Specifies the height of the Tektronix window in pixels.

fontLarge (class **Font**)
Specifies the large font to use in the Tektronix window.

font2 (class **Font**)
Specifies font number 2 to use in the Tektronix window.

font3 (class **Font**)
Specifies font number 2 font to use in the Tektronix window.

fontSmall (class **Font**)
Specifies the small font to use in the Tektronix window.

The resources that may be specified for the various menus are described in the documentation for the Athena **SimpleMenu** widget. The name and classes of the entries in each of the menus are listed below.

The *mainMenu* has the following entries:

securekbd (class **SmeBSB**)
This entry invokes the **secure()** action.

allowsends (class **SmeBSB**)
This entry invokes the **allow-send-events(toggle)** action.

logging (class **SmeBSB**)
This entry invokes the **set-logging(toggle)** action.

redraw (class **SmeBSB**)
This entry invokes the **redraw()** action.

line1 (class **SmeLine**)
This is a separator.

suspend (class **SmeBSB**)
This entry invokes the **send-signal(suspend)** action on systems that support job control.

continue (class **SmeBSB**)
This entry invokes the **send-signal(cont)** action on systems that support job control.

interrupt (class **SmeBSB**)
This entry invokes the **send-signal(int)** action.

hangup (class **SmeBSB**)
This entry invokes the **send-signal(hup)** action.

terminate (class **SmeBSB**)
This entry invokes the **send-signal(term)** action.

kill (class **SmeBSB**)
This entry invokes the **send-signal(kill)** action.

line2 (class **SmeLine**)
This is a separator.

quit (class **SmeBSB**)
This entry invokes the **quit()** action.

The *vtMenu* has the following entries:

scrollbar (class **SmeBSB**)
This entry invokes the **set-scrollbar(toggle)** action.

jumpscroll (class **SmeBSB**)
This entry invokes the **set-jumpscroll(toggle)** action.

reversevideo (class **SmeBSB**)
This entry invokes the **set-reverse-video(toggle)** action.

autowrap (class **SmeBSB**)
This entry invokes the **set-autowrap(toggle)** action.

reversewrap (class **SmeBSB**)
This entry invokes the **set-reversewrap(toggle)** action.

autolinefeed (class **SmeBSB**)
This entry invokes the **set-autolinefeed(toggle)** action.

appcursor (class **SmeBSB**)
This entry invokes the **set-appcursor(toggle)** action.

appkeypad (class **SmeBSB**)
This entry invokes the **set-appkeypad(toggle)** action.

scrollkey (class **SmeBSB**)
This entry invokes the **set-scroll-on-key(toggle)** action.

scrollttyoutput (class **SmeBSB**)
This entry invokes the **set-scroll-on-tty-output(toggle)** action.

allow132 (class **SmeBSB**)
This entry invokes the **set-allow132(toggle)** action.

cursesemul (class **SmeBSB**)
This entry invokes the **set-cursesemul(toggle)** action.

visualbell (class **SmeBSB**)
This entry invokes the **set-visualbell(toggle)** action.

marginbell (class **SmeBSB**)
This entry invokes the **set-marginbell(toggle)** action.

altscreen (class **SmeBSB**)
This entry is currently disabled.

line1 (class **SmeLine**)
This is a separator.

softreset (class **SmeBSB**)
This entry invokes the **soft-reset()** action.

hardreset (class **SmeBSB**)
This entry invokes the **hard-reset()** action.

line2 (class **SmeLine**)
This is a separator.

tekshow (class **SmeBSB**)
This entry invokes the **set-visibility(tek,toggle)** action.

tekmode (class **SmeBSB**)
This entry invokes the **set-terminal-type(tek)** action.

vthide (class **SmeBSB**)
This entry invokes the **set-visibility(vt,off)** action.

The *fontMenu* has the following entries:

fontdefault (class **SmeBSB**)
This entry invokes the **set-vt-font(d)** action.

font1 (class **SmeBSB**)
This entry invokes the **set-vt-font(1)** action.

font2 (class **SmeBSB**)
This entry invokes the **set-vt-font(2)** action.

font3 (class **SmeBSB**)
This entry invokes the **set-vt-font(3)** action.

font4 (class **SmeBSB**)
This entry invokes the **set-vt-font(4)** action.

fontescape (class **SmeBSB**)
This entry invokes the **set-vt-font(e)** action.

fontsel (class **SmeBSB**)
This entry invokes the **set-vt-font(s)** action.

The *tekMenu* has the following entries:

tektextlarge (class **SmeBSB**)
This entry invokes the **set-tek-text(l)** action.

tektext2 (class **SmeBSB**)
This entry invokes the **set-tek-text(2)** action.

tektext3 (class **SmeBSB**)
This entry invokes the **set-tek-text(3)** action.

tektextsmall (class **SmeBSB**)
This entry invokes the **set-tek-text(s)** action.

line1 (class **SmeLine**)
This is a separator.

tekpage (class **SmeBSB**)
This entry invokes the **tek-page()** action.

tekreset (class **SmeBSB**)
This entry invokes the **tek-reset()** action.

tekcopy (class **SmeBSB**)
This entry invokes the **tek-copy()** action.

line2 (class **SmeLine**)
This is a separator.

vtshow (class **SmeBSB**)
This entry invokes the **set-visibility(vt,toggle)** action.

vtmode (class **SmeBSB**)
This entry invokes the **set-terminal-type(vt)** action.

tekhide (class **SmeBSB**)
This entry invokes the **set-visibility(tek,toggle)** action.

The following resources are useful when specified for the Athena Scrollbar widget:

thickness (class **Thickness**)
Specifies the width in pixels of the scrollbar.

background (class **Background**)
Specifies the color to use for the background of the scrollbar.

foreground (class **Foreground**)
Specifies the color to use for the foreground of the scrollbar. The ''thumb'' of the scrollbar is a simple checkerboard pattern alternating pixels for foreground and background color.

EMULATIONS

The VT102 emulation is fairly complete, but does not support the blinking character attribute nor the double-wide and double-size character sets. *Termcap*(5) entries that work with *xterm* include ''xterm'', ''vt102'', ''vt100'' and ''ansi'', and *xterm* automatically searches the termcap file in this order for these entries and then sets the ''TERM'' and the ''TERMCAP'' environment variables.

Many of the special *xterm* features (like logging) may be modified under program control through a set of escape sequences different from the standard VT102 escape sequences. (See the *''Xterm Control Sequences''* document.)

The Tektronix 4014 emulation is also fairly good. Four different font sizes and five different lines types are supported. The Tektronix text and graphics commands are recorded internally by *xterm* and may be written to a file by sending the COPY escape sequence (or through the **Tektronix** menu; see below). The name of the file will be ''**COPY***yy–MM–dd.hh:mm:ss*'', where *yy*, *MM*, *dd*, *hh*, *mm* and *ss* are the year, month, day, hour, minute and second when the COPY was performed (the file is created in the directory *xterm* is started in, or the home directory for a login *xterm*).

POINTER USAGE

Once the VT102 window is created, *xterm* allows you to select text and copy it within the same or other windows.

The selection functions are invoked when the pointer buttons are used with no modifiers, and when they are used with the ''shift'' key. The assignment of the functions described below to keys and buttons may be changed through the resource database; see **ACTIONS** below.

Pointer button one (usually left) is used to save text into the cut buffer. Move the cursor to beginning of the text, and then hold the button down while moving the cursor to the end of the region and releasing the button. The selected text is highlighted and is saved in the global cut buffer and made the PRIMARY selection when the button is released. Double-clicking selects by words. Triple-clicking selects by lines. Quadruple-clicking goes back to characters, etc. Multiple-click is determined by the time from button up to button down, so you can change the selection unit in the middle of a selection. If the key/button bindings specify that an X selection is to be made, *xterm* will leave the selected text highlighted for as long as it is the selection owner.

Pointer button two (usually middle) 'types' (pastes) the text from the PRIMARY selection, if any, otherwise from the cut buffer, inserting it as keyboard input.

Pointer button three (usually right) extends the current selection. (Without loss of generality, that is you can swap ''right'' and ''left'' everywhere in the rest of this paragraph...) If pressed while closer to the right edge of the selection than the left, it extends/contracts the right edge of the selection. If you contract the selection past the left edge of the selection, *xterm* assumes you really meant the left edge, restores the original selection, then extends/contracts the left edge of the selection. Extension starts in the selection unit mode that the last selection or extension was performed in; you can multiple-click to cycle through them.

By cutting and pasting pieces of text without trailing new lines, you can take text from several places in different windows and form a command to the shell, for example, or take output from a program and insert it into your favorite editor. Since the cut buffer is globally shared among different applications, you should regard it as a 'file' whose contents you know. The terminal emulator and other text programs should be treating it as if it were a text file, i.e. the text is delimited by new lines.

The scroll region displays the position and amount of text currently showing in the window (highlighted) relative to the amount of text actually saved. As more text is saved (up to the maximum), the size of the highlighted area decreases.

Clicking button one with the pointer in the scroll region moves the adjacent line to the top of the display window.

Clicking button three moves the top line of the display window down to the pointer position.

Clicking button two moves the display to a position in the saved text that corresponds to the pointer's position in the scrollbar.

Unlike the VT102 window, the Tektronix window dows not allow the copying of text. It does allow Tektronix GIN mode, and in this mode the cursor will change from an arrow to a cross. Pressing any key will send that key and the current coordinate of the cross cursor. Pressing button one, two, or three will return the letters 'l', 'm', and 'r', respectively. If the 'shift' key is pressed when a pointer button is pressed, the corresponding upper case letter is sent. To distinguish a pointer button from a key, the high bit of the character is set (but this is bit is normally stripped unless the terminal mode is RAW; see *tty*(4) for details).

MENUS

Xterm has four menus, named *mainMenu*, *vtMenu*, *fontMenu*, and *tekMenu*. Each menu pops up under the correct combinations of key and button presses. Most menus are divided into two section, separated by a horizontal line. The top portion contains various modes that can be altered. A check mark appears next to a mode that is currently active. Selecting one of these modes toggles its state. The bottom portion of the menu are command entries; selecting one of these performs the indicated function.

The **xterm** menu pops up when the "control" key and pointer button one are pressed in a window. The *mainMenu* contains items that apply to both the VT102 and Tektronix windows. The **Secure Keyboard** mode is be used when typing in passwords or other sensitive data in an unsecure environment; see **SECURITY** below. Notable entries in the command section of the menu are the **Continue**, **Suspend**, **Interrupt**, **Hangup**, **Terminate** and **Kill** which sends the SIGCONT, SIGTSTP, SIGINT, SIGHUP, SIGTERM and SIGKILL signals, respectively, to the process group of the process running under *xterm* (usually the shell). The **Continue** function is especially useful if the user has accidentally typed CTRL-Z, suspending the process.

The *vtMenu* sets various modes in the VT102 emulation, and is popped up when the "control" key and pointer button two are pressed in the VT102 window. In the command section of this menu, the soft reset entry will reset scroll regions. This can be convenient when some program has left the scroll regions set incorrectly (often a problem when using VMS or TOPS-20). The full reset entry will clear the screen, reset tabs to every eight columns, and reset the terminal modes (such as wrap and smooth scroll) to their initial states just after *xterm* has finished processing the command line options.

The *fontMenu* sets the font used in the VT102 window.

The *tekMenu* sets various modes in the Tektronix emulation, and is popped up when the ''control'' key and pointer button two are pressed in the Tektronix window. The current font size is checked in the modes section of the menu. The **PAGE** entry in the command section clears the Tektronix window.

SECURITY

X environments differ in their security consciousness. MIT servers, run under *xdm*, are capable of using a ''magic cookie'' authorization scheme that can provide a reasonable level of security for many people. If your server is only using a host-based mechanism to control access to the server (see *xhost(1)*), then if you enable access for a host and other users are also permitted to run clients on that same host, there is every possibility that someone can run an application that will use the basic services of the X protocol to snoop on your activities, potentially capturing a transcript of everything you type at the keyboard. This is of particular concern when you want to type in a password or other sensitive data. The best solution to this problem is to use a better authorization mechanism that host-based control, but a simple mechanism exists for protecting keyboard input in *xterm*.

The **xterm** menu (see **MENUS** above) contains a **Secure Keyboard** entry which, when enabled, ensures that all keyboard input is directed *only* to *xterm* (using the GrabKeyboard protocol request). When an application prompts you for a password (or other sensitive data), you can enable **Secure Keyboard** using the menu, type in the data, and then disable **Secure Keyboard** using the menu again. Only one X client at a time can secure the keyboard, so when you attempt to enable **Secure Keyboard** it may fail. In this case, the bell will sound. If the **Secure Keyboard** succeeds, the foreground and background colors will be exchanged (as if you selected the **Reverse Video** entry in the **Modes** menu); they will be exchanged again when you exit secure mode. If the colors do *not* switch, then you should be *very* suspicious that you are being spoofed. If the application you are running displays a prompt before asking for the password, it is safest to enter secure mode *before* the prompt gets displayed, and to make sure that the prompt gets displayed correctly (in the new colors), to minimize the probability of spoofing. You can also bring up the menu again and make sure that a check mark appears next to the entry.

Secure Keyboard mode will be disabled automatically if your xterm window becomes iconified (or otherwise unmapped), or if you start up a reparenting window manager (that places a title bar or other decoration around the window) while in **Secure Keyboard** mode. (This is a feature of the X protocol not easily overcome.) When this happens, the foreground and background colors will be switched back and the bell will sound in warning.

CHARACTER CLASSES

Clicking the middle mouse button twice in rapid succession will cause all characters of the same class (e.g. letters, white space, punctuation) to be selected. Since different people have different preferences for what should be selected (for example, should filenames be selected as a whole or only the separate subnames), the default mapping can be overridden through the use of the *charClass* (class *CharClass*) resource.

This resource is simply a list of *range:value* pairs where the range is either a single number or *low-high* in the range of 0 to 127, corresponding to the ASCII code for the character or characters to be set. The *value* is arbitrary, although the default table uses the character number of the first character occurring in the set.

The default table is:

```
static int charClass[128] = {
/* NUL SOH STX ETX EOT ENQ ACK BEL */
  32,  1,  1,  1,  1,  1,  1,  1,
/* BS  HT  NL  VT  NP  CR  SO  SI */
   1, 32,  1,  1,  1,  1,  1,  1,
/* DLE DC1 DC2 DC3 DC4 NAK SYN ETB */
   1,  1,  1,  1,  1,  1,  1,  1,
/* CAN EM  SUB ESC FS  GS  RS  US */
   1,  1,  1,  1,  1,  1,  1,  1,
/* SP  !   "   #   $   %   &   ' */
  32, 33, 34, 35, 36, 37, 38, 39,
/* (   )   *   +   ,   -   .   / */
  40, 41, 42, 43, 44, 45, 46, 47,
/* 0   1   2   3   4   5   6   7 */
  48, 48, 48, 48, 48, 48, 48, 48,
/* 8   9   :   ;   <   =   >   ? */
  48, 48, 58, 59, 60, 61, 62, 63,
/* @   A   B   C   D   E   F   G */
  64, 48, 48, 48, 48, 48, 48, 48,
/* H   I   J   K   L   M   N   O */
  48, 48, 48, 48, 48, 48, 48, 48,
/* P   Q   R   S   T   U   V   W */
  48, 48, 48, 48, 48, 48, 48, 48,
/* X   Y   Z   [   \   ]   ^   _ */
  48, 48, 48, 91, 92, 93, 94, 48,
/* `   a   b   c   d   e   f   g*/
  96, 48, 48, 48, 48, 48, 48, 48,
/* h   i   j   k   l   m   n   o*/
  48, 48, 48, 48, 48, 48, 48, 48,
/* p   q   r   s   t   u   v   w*/
  48, 48, 48, 48, 48, 48, 48, 48,
/* x   y   z   {   |   }   ~ DEL */
  48, 48, 48,123,124,125,126,  1};
```

For example, the string "33:48,37:48,45-47:48,64:48" indicates that the exclamation mark, percent sign, dash, period, slash, and ampersand characters should be treated the same way as characters and numbers. This is very useful for cutting and pasting electronic mailing addresses and filenames.

ACTIONS

It is possible to rebind keys (or sequences of keys) to arbitrary strings for input, by changing the translations for the vt100 or tek4014 widgets. Changing the translations for events other than key and button events is not expected, and will cause unpredictable behavior. The following actions are provided for using within the *vt100* or *tek4014* **translations** resources:

bell([*percent*]**)**
This action rings the keyboard bell at the specified percentage above or below the base volume.

ignore() This action ignores the event but checks for special pointer position escape sequences.

insert() This action is a synonym for **insert-seven-bit()**

insert-seven-bit()
This action inserts the 7-bit USASCII character or string associated with the keysym that was pressed.

insert-eight-bit()
This action inserts the 8-bit ISO Latin-1 character or string associated with the keysym that was pressed.

insert-selection(*sourcename* **[, ...])**
This action inserts the string found in the selection or cutbuffer indicated by *sourcename*. Sources are checked in the order given (case is significant) until one is found. Commonly-used selections include: *PRIMARY*, *SECONDARY*, and *CLIPBOARD*. Cut buffers are typically named *CUT_BUFFER0* through *CUT_BUFFER7*.

keymap(*name*)
This action dynamically defines a new translation table whose resource name is *name* with the suffix *Keymap* (case is significant). The name *None* restores the original translation table.

popup-menu(*menuname*)
This action displays the specified popup menu. Valid names (case is significant) include: *mainMenu*, *vtMenu*, *fontMenu*, and *tekMenu*.

secure() This action toggles the *Secure Keyboard* mode described in the section named **SECURITY**, and is invoked from the **securekbd** entry in *mainMenu*.

select-start()
This action begins text selection at the current pointer location. See the section on **POINTER USAGE** for information on making selections.

select-extend()
This action tracks the pointer and extends the selection. It should only be bound to Motion events.

select-end(*destname* [, ...])
This action puts the currently selected text into all of the selections or cutbuffers specified by *destname*.

select-cursor-start()
This action is similar to **select-start** except that it begins the selection at the current text cursor position.

select-cursor-end(*destname* [, ...])
This action is similar to **select-end** except that it should be used with **select-cursor-start**.

set-vt-font(*d/1/2/3/4/e/s* [,*normalfont* [, *boldfont*]])
This action sets the font or fonts currently being used in the VT102 window. The first argument is a single character that specifies the font to be used: *d* or *D* indicate the default font (the font initially used when *xterm* was started), *1* through *4* indicate the fonts specified by the *font1* through *font4* resources, *e* or *E* indicate the normal and bold fonts that may be set through escape codes (or specified as the second and third action arguments, respectively), and *i* or *I* indicate the font selection (as made by programs such as *xfontsel(1)*) indicated by the second action argument.

start-extend()
: This action is similar to **select-start except that the selection is extended to the current pointer location.**

start-cursor-extend()
: This action is similar to **select-extend** except that the selection is extended to the current text cursor position.

string(*string***)**
: This action inserts the specified text string as if it had been typed. Quotation is necessary if the string contains whitespace or non-alphanumeric characters. If the string argument begins with the characters ''0x'', it is interpreted as a hex character constant.

scroll-back(*count* **[,***units***])**
: This action scrolls the text window backward so that text that had previously scrolled off the top of the screen is now visible. The *count* argument indicates the number of *units* (which may be *page*, *halfpage*, *pixel*, or *line*) by which to scroll.

scroll-forw(*count* **[,***units***])**
: This action scrolls is similar to **scroll-back** except that it scrolls the other direction.

allow-send-events(*on/off/toggle***)**
: This action set or toggles the **allowSendEvents** resource and is also invoked by the **allowsends** entry in *mainMenu.*

set-logging(*on/off/toggle***)**
: This action toggles the **logging** resource and is also invoked by the **logging** entry in *mainMenu.*

redraw()
: This action redraws the window and is also invoked by the *redraw* entry in *mainMenu.*

send-signal(*signame***)**
: This action sends the signal named by *signame* (which may also be a number) to the *xterm* subprocess (the shell or program specified with the *-e* command line option) and is also invoked by the **suspend**, **continue**, **interrupt**, **hangup**, **terminate**, and *kill* entries in *mainMenu.* Allowable signal names are (case is not significant): *suspend*, *tstp* (if supported by the operating system), *cont* (if supported by the operating system), *int*, *hup*, *term*, and *kill.*

quit() This action sends a SIGHUP to the subprogram and exits. It is also invoked by the **quit** entry in *mainMenu.*

set-scrollbar(*on/off/toggle***)**
This action toggles the **scrollbar** resource and is also invoked by the **scrollbar** entry in *vtMenu.*

set-jumpscroll(*on/off/toggle***)**
This action toggles the **jumpscroll** resource and is also invoked by the **jumpscroll** entry in *vtMenu.*

set-reverse-video(*on/off/toggle***)**
This action toggles the *reverseVideo* resource and is also invoked by the **reversevideo** entry in *vtMenu.*

set-autowrap(*on/off/toggle***)**
This action toggles automatic wrapping of long lines and is also invoked by the **autowrap** entry in *vtMenu.*

set-reversewrap(*on/off/toggle***)**
This action toggles the **reverseWrap** resource and is also invoked by the **reversewrap** entry in *vtMenu.*

set-autolinefeed(*on/off/toggle***)**
This action toggles automatic insertion of linefeeds and is also invoked by the **autolinefeed** entry in *vtMenu.*

set-appcursor(*on/off/toggle***)**
This action toggles the handling Application Cursor Key mode and is also invoked by the Bappcursor entry in *vtMenu.*

set-appkeypad(*on/off/toggle***)**
This action toggles the handling of Application Keypad mode and is also invoked by the **appkeypad** entry in *vtMenu.*

set-scroll-on-key(*on/off/toggle***)**
This action toggles the **scrollKey** resource and is also invoked from the **scrollkey** entry in *vtMenu.*

set-scroll-on-tty-output(*on/off/toggle***)**
This action toggles the **scrollTtyOutput** resource and is also invoked from the **scrollttyoutput** entry in *vtMenu.*

set-allow132(*on/off/toggle***)**
This action toggles the **c132** resource and is also invoked from the **allow132** entry in *vtMenu.*

set-cursesemul(*on/off/toggle***)**
This action toggles the **curses** resource and is also invoked from the **cursesemul** entry in *vtMenu.*

set-visual-bell(*on/off/toggle***)**
This action toggles the **visualBell** resource and is also invoked by the **visualbell** entry in *vtMenu.*

set-marginbell(*on/off/toggle***)**
This action toggles the **marginBell** resource and is also invoked from the **marginbell** entry in *vtMenu.*

set-altscreen(*on/off/toggle***)**
This action toggles between the alternative and current screens.

soft-reset()
This action resets the scrolling region and is also invoked from the **softreset** entry in *vtMenu.*

hard-reset()
This action resets the scrolling region, tabs, window size, and cursor keys and clears the screen. It is also invoked from the **hardreset** entry in *vtMenu.*

set-terminal-type(*type***)**
This action directs output to either the *vt* or *tek* windows, according to the *type* string. It is also invoked by the **tekmode** entry in *vtMenu* and the **vtmode** entry in *tekMenu.*

set-visibility(*vt/tek,on/off/toggle***)**
This action controls whether or not the *vt* or *tek* windows are visible. It is also invoked from the **tekshow** and **vthide** entries in *vtMenu* and the **vtshow** and **tekhide** entries in *tekMenu.*

set-tek-text(*large/2/3/small***)**
This action sets font used in the Tektronix window to the value of the resources **tektextlarge, tektext2, tektext3,** and **tektextsmall** according to the argument. It is also by the entries of the same names as the resources in *tekMenu.*

tek-page()
This action clears the Tektronix window and is also invoked by the **tekpage** entry in *tekMenu.*

tek-reset()
This action resets the Tektronix window and is also invoked by the *tekreset* entry in *tekMenu.*

tek-copy()
This action copies the escape codes used to generate the current window contents to a file in the current directory beginning with the name COPY. It is also invoked from the *tekcopy* entry in *tekMenu.*

gin-press(*l*/*L*/*m*/*M*/*r*/*R*)
This action send the indicated graphics input code.

The default bindings in the VT102 window are:

Shift <KeyPress> Prior:	scroll-back(1,halfpage)
Shift <KeyPress> Next:	scroll-forw(1,halfpage)
Shift <KeyPress> Select:	select-cursor-start() select-cursor-end(PRIMARY, CUT_BUFFER0)
Shift <KeyPress> Insert:	insert-selection(PRIMARY, CUT_BUFFER0)
~Meta<KeyPress>:	insert-seven-bit()
Meta<KeyPress>:	insert-eight-bit()
Ctrl ~Meta<Btn1Down>:	popup-menu(mainMenu)
~Meta <Btn1Down>:	select-start()
~Meta <Btn1Motion>:	select-extend()
Ctrl ~Meta <Btn2Down>:	popup-menu(vtMenu)
~Ctrl ~Meta <Btn2Down>:	ignore()
~Ctrl ~Meta <Btn2Up>:	insert-selection(PRIMARY, CUT_BUFFER0)
Ctrl ~Meta <Btn3Down>:	popup-menu(fontMenu)
~Ctrl ~Meta <Btn3Down>:	start-extend()
~Meta <Btn3Motion>:	select-extend()
~Ctrl ~Meta <BtnUp>:	select-end(PRIMARY, CUT_BUFFER0)
<BtnDown>:	bell(0)

The default bindings in the Tektronix window are:

~Meta<KeyPress>:	insert-seven-bit()
Meta<KeyPress>:	insert-eight-bit()
Ctrl ~Meta<Btn1Down>:	popup-menu(mainMenu)
Ctrl ~Meta <Btn2Down>:	popup-menu(tekMenu)
Shift ~Meta<Btn1Down>:	gin-press(L)
~Meta<Btn1Down>:	gin-press(l)
Shift ~Meta<Btn2Down>:	gin-press(M)
~Meta<Btn2Down>:	gin-press(m)
Shift ~Meta<Btn3Down>:	gin-press(R)
~Meta<Btn3Down>:	gin-press(r)

Below is a sample how of the **keymap()** action is used to add special keys for entering commonly-typed works:

```
*VT100.Translations: #override <Key>F13: keymap(dbx)
*VT100.dbxKeymap.translations: \
  <Key>F14: keymap(None) \n\
  <Key>F17: string("next") string(0x0d) \n\
  <Key>F18: string("step") string(0x0d) \n\
  <Key>F19: string("continue") string(0x0d) \n\
  <Key>F20: string("print ") insert-selection(PRIMARY, CUT_BUFFER0)
```

OTHER FEATURES

Xterm automatically highlights the window border and text cursor when the pointer enters the window (selected) and unhighlights them when the pointer leaves the window (unselected). If the window is the focus window, then the window is highlighted no matter where the pointer is.

In VT102 mode, there are escape sequences to activate and deactivate an alternate screen buffer, which is the same size as the display area of the window. When activated, the current screen is saved and replace with the alternate screen. Saving of lines scrolled off the top of the window is disabled until the normal screen is restored. The *termcap*(5) entry for *xterm* allows the visual editor *vi*(1) to switch to the alternate screen for editing, and restore the screen on exit.

In either VT102 or Tektronix mode, there are escape sequences to change the name of the windows and to specify a new log file name.

ENVIRONMENT

Xterm sets the environment variables ''TERM'' and ''TERMCAP'' properly for the size window you have created. It also uses and sets the environment variable ''DISPLAY'' to specify which bit map display terminal to use. The environment variable ''WINDOWID'' is set to the X window id number of the *xterm* window.

SEE ALSO

resize(1), X(1), pty(4), tty(4)
Xterm Control Sequences

BUGS

The *Xterm Control Sequences* document has yet to be converted from X10. The old version, along with a first stab at an update, are available in the sources.

The class name is *XTerm* instead of *Xterm*.

Xterm will hang forever if you try to paste too much text at one time. It is both producer and consumer for the pty and can deadlock.

Variable-width fonts are not handled.

This program still needs to be rewritten. It should be split into very modular sections, with the various emulators being completely separate widgets that don't know about each other. Ideally, you'd like to be able to pick and choose emulator widgets and stick them into a single control widget.

The focus is considered lost if some other client (e.g., the window manager) grabs the pointer; it is difficult to do better without an addition to the protocol.

There needs to be a dialog box to allow entry of log file name and the COPY file name.

Many of the options are not resettable after *xterm* starts.

The Tek widget does not support key/button re-binding.

COPYRIGHT

AUTHORS

Far too many people, including:

Loretta Guarino Reid (DEC-UEG-WSL), Joel McCormack (DEC-UEG-WSL), Terry Weissman (DEC-UEG-WSL), Edward Moy (Berkeley), Ralph R. Swick (MIT-Athena), Mark Vandevoorde (MIT-Athena), Bob McNamara (DEC-MAD), Jim Gettys (MIT-Athena), Bob Scheifler (MIT X Consortium), Doug Mink (SAO), Steve Pitschke (Stellar), Ron Newman (MIT-Athena), Jim Fulton (MIT X Consortium), Dave Serisky (HP)

NAME

xwd - dump an image of an X window

SYNOPSIS

xwd [-debug] [-help] [-nobdrs] [-out *file*] [-xy] [-frame] [-display *display*]

DESCRIPTION

Xwd is an X Window System window dumping utility. *Xwd* allows X users to store window images in a specially formatted dump file. This file can then be read by various other X utilities for redisplay, printing, editing, formatting, archiving, image processing, etc. The target window is selected by clicking the mouse in the desired window. The keyboard bell is rung once at the beginning of the dump and twice when the dump is completed.

OPTIONS

-display *display*
: This argument allows you to specify the server to connect to; see *X(1)*.

-help Print out the 'Usage:' command syntax summary.

-nobdrs This argument specifies that the window dump should not include the pixels that compose the X window border. This is useful in situations where you may wish to include the window contents in a document as an illustration.

-out *file* This argument allows the user to explicitly specify the output file on the command line. The default is to output to standard out.

-xy This option applies to color displays only. It selects 'XY' format dumping instead of the default 'Z' format.

-add *value*
: This option specifies an signed value to be added to every pixel.

-frame This option indicates that the window manager frame should be included when manually selecting a window.

ENVIRONMENT

DISPLAY
: To get default host and display number.

FILES

XWDFile.h
: X Window Dump File format definition file.

SEE ALSO

xwud(1), xpr(1), X(1)

COPYRIGHT

AUTHORS

Tony Della Fera, Digital Equipment Corp., MIT Project Athena
William F. Wyatt, Smithsonian Astrophysical Observatory

NAME

xwininfo - window information utility for X

SYNOPSIS

xwininfo [-help] [-id *id*] [-root] [-name *name*] [-int] [-tree] [-stats] [-bits] [-events] [-size] [-wm] [-frame] [-all] [-english] [-metric] [-display *display*]

DESCRIPTION

Xwininfo is a utility for displaying information about windows. Various information is displayed depending on which options are selected. If no options are chosen, **-stats** is assumed.

The user has the option of selecting the target window with the mouse (by clicking any mouse button in the desired window) or by specifying its window id on the command line with the **-id** option. Or instead of specifying the window by its id number, the **-name** option may be used to specify which window is desired by name. There is also a special **-root** option to quickly obtain information on X's root window.

OPTIONS

-help Print out the 'Usage:' command syntax summary.

-id *id* This option allows the user to specify a target window *id* on the command line rather than using the mouse to select the target window. This is very useful in debugging X applications where the target window is not mapped to the screen or where the use of the mouse might be impossible or interfere with the application.

-name *name*
This option allows the user to specify that the window named *name* is the target window on the command line rather than using the mouse to select the target window.

-root This option specifies that X's root window is the target window. This is useful in situations where the root window is completely obscured.

-int This option specifies that all X window ids should be displayed as integer values. The default is to display them as hexadecimal values.

-tree This option causes the root, parent, and children windows' ids and names of the selected window to be displayed.

-stats This option causes the display of various attributes pertaining to the location and appearance of the selected window. Information displayed includes the location of the window, its width and height, its depth, border width, class, colormap id if any, map state, backing-store hint, and location of the corners.

-bits This option causes the display of various attributes pertaining to the selected window's raw bits and how the selected window is to be stored. Displayed information includes the selected window's bit gravity, window gravity, backing-store hint, backing-planes value, backing pixel, and whether or not the window has save-under set.

-events This option causes the selected window's event masks to be displayed. Both the event mask of events wanted by some client and the event mask of events not to propagate are displayed.

-size This option causes the selected window's sizing hints to be displayed. Displayed information includes: for both the normal size hints and the zoom size hints, the user supplied location if any; the program supplied location if any; the user supplied size if any; the program supplied size if any; the minimum size if any; the maximum size if any; the resize increments if any; and the minimum and maximum aspect ratios if any.

-wm This option causes the selected window's window manager hints to be displayed. Information displayed may include whether or not the application accepts input, what the window's icon window # and name is, where the window's icon should go, and what the window's initial state should be.

-frame This option causes window manager frames not be ignored when manually selecting windows.

-metric This option causes all individual height, width, and x and y positions to be displayed in millimeters as well as number of pixels, based on what the server thinks the resolution is. Geometry specifications that are in +x+y form are not changed.

-english This option causes all individual height, width, and x and y positions to be displayed in inches (and feet, yards, and miles if necessary) as well as number of pixels. **-metric** and **-english** may both be enabled at the same time.

-all This option is a quick way to ask for all information possible.

-display *display*
This option allows you to specify the server to connect to; see *X(1)*.

EXAMPLE

The following is a sample summary taken with no options specified:

```
xwininfo ==> Please select the window about which you
     ==> would like information by clicking the
```

```
          ==> mouse in that window.

      xwininfo ==> Window id: 0x60000f (xterm)

          ==> Upper left X: 4
          ==> Upper left Y: 19
          ==> Width: 726
          ==> Height: 966
          ==> Depth: 4
          ==> Border width: 3
          ==> Window class: InputOutput
          ==> Colormap: 0x80065
          ==> Window Bit Gravity State: NorthWestGravity
          ==> Window Window Gravity State: NorthWestGravity
          ==> Window Backing Store State: NotUseful
          ==> Window Save Under State: no
          ==> Window Map State: IsViewable
          ==> Window Override Redirect State: no
          ==> Corners:  +4+19  -640+19  -640-33  +4-33
```

ENVIRONMENT

DISPLAY

To get the default host and display number.

SEE ALSO

X(1), xprop(1)

BUGS

Using **-stats -bits** shows some redundant information.

COPYRIGHT

Copyright 1988, Massachusetts Institute of Technology.
See *X(1)* for a full statement of rights and permissions.

AUTHOR

Mark Lillibridge, MIT Project Athena

NAME

xwud - image displayer for X

SYNOPSIS

xwud [-in *file*] [-noclick] [-geometry *geom*] [-display *display*] [-new] [-std <maptype>] [-raw] [-vis <vis-type-or-id>] [-help] [-rv] [-plane *number*] [-fg *color*] [-bg *color*]

DESCRIPTION

Xwud is an X Window System image undumping utility. *Xwud* allows X users to display in a window an image saved in a specially formatted dump file, such as produced by *xwd(1)*.

OPTIONS

-bg *color*
: If a bitmap image (or a single plane of an image) is displayed, this option can be used to specify the color to display for the "0" bits in the image.

-display *display*
: This option allows you to specify the server to connect to; see *X(1)*.

-fg *color*
: If a bitmap image (or a single plane of an image) is displayed, this option can be used to specify the color to display for the "1" bits in the image.

-geometry *geom*
: This option allows you to specify the size and position of the window. Typically you will only want to specify the position, and let the size default to the actual size of the image.

-help
: Print out a short description of the allowable options.

-in *file*
: This option allows the user to explicitly specify the input file on the command line. If no input file is given, the standard input is assumed.

-new
: This option forces creation of a new colormap for displaying the image. If the image characteristics happen to match those of the display, this can get the image on the screen faster, but at the cost of using a new colormap (which on most displays will cause other windows to go technicolor).

-noclick
: Clicking any button in the window will terminate the application, unless this option is specified. Termination can always be achieved by typing 'q', 'Q', or ctrl-c.

-plane *number*
You can select a single bit plane of the image to display with this option. Planes are numbered with zero being the least significant bit. This option can be used to figure out which plane to pass to *xpr(1)* for printing.

-raw This option forces the image to be displayed with whatever color values happen to currently exist on the screen. This option is mostly useful when undumping an image back onto the same screen that the image originally came from, while the original windows are still on the screen, and results in getting the image on the screen faster.

-rv If a bitmap image (or a single plane of an image) is displayed, this option forces the foreground and background colors to be swapped. This may be needed when displaying a bitmap image which has the color sense of pixel values "0" and "1" reversed from what they are on your display.

-std *maptype*
This option causes the image to be displayed using the specified Standard Colormap. The property name is obtained by converting the type to upper case, prepending "RGB_", and appending "_MAP". Typical types are "best", "default", and "gray". See *xcmap(1)* for one way of creating Standard Colormaps.

-vis *vis-type-or-id*
This option allows you to specify a particular visual or visual class. The default is to pick the "best" one. A particular class can be specified: "StaticGray", "GrayScale", "StaticColor", "PseudoColor", "DirectColor", or "TrueColor". Or "Match" can be specified, meaning use the same class as the source image. Alternatively, an exact visual id (specific to the server) can be specified, either as a hexadecimal number (prefixed with "0x") or as a decimal number. Finally, "default" can be specified, meaning to use the same class as the colormap of the root window. Case is not signficant in any of these strings.

ENVIRONMENT

DISPLAY
To get default display.

FILES

XWDFile.h
X Window Dump File format definition file.

SEE ALSO

xwd(1), xpr(1), xcmap(1), X(1)

COPYRIGHT

AUTHOR

Bob Scheifler, MIT X Consortium

NAME

yacc – yet another compiler-compiler

SYNOPSIS

yacc [**–vdlt**] grammar

DESCRIPTION

The *yacc* command converts a context-free grammar into a set of tables for a simple automaton which executes an *LR*(1) parsing algorithm. The grammar may be ambiguous; specified precedence rules are used to break ambiguities.

The output file, **y.tab.c**, must be compiled by the C compiler to produce a program *yyparse*. This program must be loaded with the lexical analyzer program, *yylex*, as well as *main* and *yyerror*, an error handling routine. These routines must be supplied by the user; *lex*(1) is useful for creating lexical analyzers usable by *yacc*.

If the **–v** flag is given, the file **y.output** is prepared, which contains a description of the parsing tables and a report on conflicts generated by ambiguities in the grammar.

If the **–d** flag is used, the file **y.tab.h** is generated with the **#define** statements that associate the *yacc*-assigned "token codes" with the user-declared "token names". This allows source files other than **y.tab.c** to access the token codes.

If the **–l** flag is given, the code produced in **y.tab.c** will *not* contain any **#line** constructs. This should only be used after the grammar and the associated actions are fully debugged.

Runtime debugging code is always generated in **y.tab.c** under conditional compilation control. By default, this code is not included when **y.tab.c** is compiled. However, when *yacc*'s **–t** option is used, this debugging code will be compiled by default. Independent of whether the **–t** option was used, the runtime debugging code is under the control of **YYDEBUG**, a preprocessor symbol. If **YYDEBUG** has a non-zero value, then the debugging code is included. If its value is zero, then the code will not be included. The size and execution time of a program produced without the runtime debugging code will be smaller and slightly faster.

FILES

y.output	
y.tab.c	
y.tab.h	defines for token names
yacc.tmp,	
yacc.debug, yacc.acts	temporary files
/usr/lib/yaccpar	parser prototype for C programs

SEE ALSO

lex(1).
Programmer's Guide.

DIAGNOSTICS

The number of reduce-reduce and shift-reduce conflicts is reported on the standard error output; a more detailed report is found in the **y.output** file. Similarly, if some rules are not reachable from the start symbol, this is also reported.

CAVEAT

Because file names are fixed, at most one *yacc* process can be active in a given directory at a given time.

NAME

intro – introduction to games and demos

DESCRIPTION

The manual pages with the section 6 suffix are for recreational and educational programs normally found in the directory **/usr/games**; however, Silicon Graphics currently does not ship any such programs, so this directory does not exist.

The manual pages with the suffix 6D are the demonstration programs found in the directory **/usr/demos/bin.**

The availability of these programs and the data they use (stored in the **/usr/demos/data/** directories) will vary from system to system and from release to release.

SEE ALSO

buttonfly(6D).

NAME

arena – a future sport

SYNOPSIS

arena [-n] [-d]

DESCRIPTION

It is the year 2053. The corporate wars are over, and the once great corporate empires lie in ruin. Man, however, never tired of violence, has changed the war machines of the past into today's sport. MECHS, fourteen foot tall humanoid fighting machines, and their pilots battle it out in the arenas.

arena simulates this possible future sport.

The opening screen contains instructions on how to play. Please note that you must get used to the controls because they are very sensitive. Use the left mouse button to aim before firing and the middle mouse button for sharp turns in the maze.

The **-n** option runs arena over the network using udp broadcast. This allows for a game with many players.

The **-d** option puts dummy MECHS in the arena. This gives you a chance to see other MECHS if you are not running over the network.

Pressing the ESC key exits the program.

AUTHOR

Rob Mace

NAME

boing – gravitationally attractive bouncing balls

SYNOPSIS

boing

DESCRIPTION

Boing simulates elastic and inelastic collisions of spheres and multibody gravitational effects. This program demonstrates lighting, and an example of a CPU intensive and graphics intensive simulation. For each frame, collisions of each sphere with every other sphere are checked, and the gravitational effects (if activated) of each sphere on every other sphere are calculated. This program is best described as a simulation of rubber planets.

Operation

Size the window to preference (I like really big ones, personally). Rotate and zoom the scene in the usual manner: left mouse for zoom, middle mouse for rotation. The following program options are available with a popup menu:

Turn Gravity on or off

Due to an as yet uncorrected bug, I recommend the elastics walls be turned off while gravity is on, otherwise the system accumulates all sorts of energy and eventually goes to hell on fast freight.

Drop Lines

makes the spheres look like balloons. This feature is useful for giving more 3D "who's in front of who" information.

Inverse Density

will cause the large spheres to be very "light" relative to the small spheres, which become very dense. This option is neat to watch with 2 large spheres and one small one (see Redo Spheres below), and with gravity increased about 5 times.

Negative Gravity

causes all the spheres to repel each other. I recommend this option be tried with elastic walls on and elastic balls off at lower gravity.

Elastic Walls and Elastic Balls

these toggles between completely elastic and inelastic collisions.

Blink on Collision

makes a sort of 3D pinball look.

Add Spheres

Remove Spheres

Increase Gravity

Decrease Gravity

Reset View
These options all do what they say with utmost expediency.

Redo Spheres
randomly resizes and repositions the spheres and cancels the Inverse Density option.

BUGS

The system can accumulate too much energy and explode (the physics aren't quite correct).

AUTHOR

Wade Olsen

NAME

bounce – three colored lights bouncing around a scene

SYNOPSIS

bounce [**modelname.bin**]

DESCRIPTION

Bounce displays three lights (one red, one green, and one blue) bouncing off the walls of a cube-shaped room. If given an object as an argument, the object will be displayed and lighted in the room as well. The position of the mouse pointer on the screen determines the rotation of the scene. A pop-up menu lets you turn various features on and off.

FILES

/usr/demos/data/models/ is where objects can be found.

BUGS

The user interface could use improvement.

Objects are stored in 'spin' format, which could also use improvement.

HARDWARE REQUIREMENTS

Requires at least 24 bitplanes.

NAME

buttonfly – a pretty user interface for Silicon Graphics demos

SYNOPSIS

buttonfly [**menufile**]

DESCRIPTION

Buttonfly is a user reconfigurable, hierarchical, graphical menu system.

Buttonfly opens up a window and displays one or more buttons, each with its own title. Moving the mouse pointer over a button and pressing the left mouse button selects that button, and will cause it to either execute UNIX commands (such as running a program) or fly forward and flip over, exposing one or more new buttons. Each of the newly exposed buttons may also be selected to either execute UNIX commands or expose a new level of buttons, and so on.

To go back to the previous set of buttons move the mouse pointer so it isn't over any of the buttons, and press the left mouse button (i.e. select the background). The buttons will flip over backwards to bring up the previous set. Selecting the background at the top level does nothing.

Using the middle mouse button instead of the left mouse button causes the new menu to be displayed immediately, instead of flipping it in to view.

Pressing the right mouse button will bring up a different pop-up menu depending on which button the mouse pointer is over. Each menu has at least a 'Do It' entry; selecting this entry has the same effect that pressing the left mouse button would.

Pressing the space bar will toggle between a full-screen version of *buttonfly* and the window version. The full-screen version is useful for presentations to large audiences, where visibility is important.

To exit from *buttonfly*, press the escape key.

RECONFIGURING BUTTONFLY

Buttonfly's menus were designed to be simple to change with any UNIX text editor (such as *vi*). When it is run with no arguments, it looks for button descriptions in a file called '.menu' in the current directory. If it is given one argument, it will look in that file for button definitions instead.

Buttonfly uses a very simple format to describe its buttons. The format has four different kinds of lines: comments, titles, actions, and commands.

Any line beginning with a '#' character is considered a comment and is ignored.

Button titles, like comments, always start in the first column of the file, and may be anything not starting with a '#'. Titles may be a maximum of 36 characters long; *buttonfly* will automatically separate the title into a maximum of three lines of twelve characters, separating words in the title at spaces.

Action lines start with a tab and end with a newline, and may be any UNIX command. Multiple actions may be specified for a button by listing them after the button one per line, each starting with a tab. Actions may not span more than one line, but may be any length. If no actions are specified for a button, the button is displayed, but nothing will happen when it is selected.

Commands, like actions, start with a tab character. Currently, the commands recognized are '.menu.', '.popup.', '.cd.', '.color.', '.backcolor.', and '.highcolor.'. Everything else is assumed to be an action and will be executed as a UNIX command in a shell (see *sh(1)).*

The special action called '.menu.' is used to specify a button that has several buttons attached to the back of it. When *buttonfly* sees '.menu. filename' it will create a new set of buttons from the descriptions in 'filename'. If it cannot find 'filename' in the current directory, then nothing happens. If 'filename' is '-', then the new set of buttons is created from the standard output of the previous actions (see example below). Note that any 'cd' commands in an action list do not affect the directory in which *buttonfly* looks for the filenames specified in '.menu.' commands, becuase actions are executed in a sub-shell (see the '.cd.' command below).

Another special actions is '.popup.'. It is used to define a pop-up menu entry attached to a button. When *buttonfly* sees '.popop. title' it adds an entry 'title' to the button's pop-up menu. The actions following a .popup. command will be executed when that menu entry is chosen. *Buttonfly* automatically provides a 'Do It' entry, which has the same effect as pressing the left mouse button.

The '.cd.' action is used to change buttonfly's idea of the current working directory. When the button containing the .cd. command is pressed, *buttonfly* will chdir into that directory before executing that button's actions, and all buttons underneath that button will also be executed in that directory (unless they also have .cd. commands). Note that executing the regular 'cd' shell command in a button does not affect *buttonfly's* idea of the current working directory, since all button actions are executed in their own shell environment.

The command '.color.' followed by three numbers in the range 0.0 ... 1.0 specifies the red, green and blue components of a button's color, and '.highcolor' does the same for the color of the button as it is being selected. '.backcolor.' changes the color of the back of the button, which becomes the background color of the next set of buttons (displayed when the button flips over). Note that using color commands at the top of a file, before any buttons have been defined, causes the default colors to be changed for all subsequent buttons. To change the background color to red for the first button and all subsequent buttons you would specify:

.backcolor. 1.0 0.0 0.0

in the menu description file before any buttons are defined.

EXAMPLES

The following are example buttonfly .menu files, with descriptions of what they do.

This is a very simple button called 'First' that has the buttons defined in the file '.menu_second' on the back of it.

```
First
        .menu. .menu_second
```

Here are three buttons. The 'Parent' button is purple and will flip over and display the other two buttons when pressed (they will be displayed on a purple background, since that is the color on the back of the first button). The 'Demonstrate...' button changes directories and then runs a program when pressed, and 'Recurse' flips over and shows the first button again, this time on a blue background since the '.backcolor.' command changes the background color to blue.

```
#Save this in a file called '.menu'
Parent
        .color. .89 .10 .89
        .backcolor. .89 .10 .89
        .highcolor. 1.0 .5 1.0
        .menu. child

#Save the following buttons in a file called 'child'
Demonstrate GL/NeWS Interface
        cd /usr/NeWS/demo
        glnews
Recurse
        .backcolor. 0.0 0.0 1.0
        .menu. .menu
```

Here is a very complicated button, which builds a button file based on the files it finds in a directory full of files. When pressed, it builds the file and then uses .menu. to flip over, exposing entries for all the files it found. Note that this allows you to delete or add image files, and the buttons will always be accurate. It also has a pop-up menu on it giving help; the first .popup command specifies the title of the menu using the %t feature of Silicon Graphics menus.

```
Show An Image
        cd /usr/demos/data/images
#The following two lines must all be on the
# same line in the .menu file
        /bin/ls *.rgb | awk 'BEGIN{FS="."}{print "Show " $1
"\n\tipaste /usr/demos/data/images/" $0 "\n"}'
        .menu. -
        .popup. ShowImage Menu %t
        .popup. Ipaste Man Page
        wsh -c man ipaste
```

BUGS

Buttonfly will not work if the first character of a file is a TAB followed by anything other than .color., .backcolor., or .highcolor. It is safest to eliminate any blank lines from the files you create.

Buttonfly may crash if given an arbitrary file as input.

The maximum number of buttons on any level of the hierarchy is 32.

If a program fails to run, nothing happens-- no error messages or warnings are given. This can be considered either a feature or a bug.

Specifying %F, %f, %x, or %m as part of a .popup. title will give bad results, and may crash the program. Using %t to specify a title works, but any actions specified for that entry will never be executed (since it is impossible to select a title entry).

FILES

/usr/demos/.m* are the button files the demos use.

AUTHOR

Thant Tessman and Gavin Bell

HARDWARE REQUIREMENTS

Must have at least 24 bitplanes.

NAME

cedit – edit colors on the screen

SYNOPSIS

/usr/sbin/cedit

DESCRIPTION

The frame buffer of the IRIS contains values which are translated into RGB values by a color map. *cedit* changes the mapping of any color index. Three sliders are displayed, along with a sample patch of the color being edited. A click of the left button outside the screen area of *cedit* picks up a color index to be edited. The sliders indicate the red, green, and blue components of the current color. The color can be changed by clicking down on a slider, and adjusting the position of the control. Under each slider control is a number representing the current value of that color component. The number ranges from 0 (for least intensity) to 255 (for most intensity). Under the color patch is a number that represents the current color index.

SEE ALSO

showmap(6D)

BUGS

If another process changes the color entry being edited, the sliders of cedit do not indicate the correct positions for that color index.

This program must have input focus in order to work properly. See the window manager documentation for a discussion of input focus.

NAME

clock – analog clock in a window

SYNOPSIS

/usr/sbin/clock

DESCRIPTION

Clock is a window manager graphics program which shows the current (machine) time. The current machine time is set after you boot up the system, when you go into multi-user mode. If the argument "dot" is used when invoking the clock (typing "clock dot"), a small, decorative dot covers the middle of the clock face.

Selecting the Calendar menu button invokes the cal (1) program.

Selecting the Alarm menu button causes the alarm hour hand to be displayed in red, and the alarm time to be displayed at the bottom of the clock. The right mouse button sets the alarm time, based on the angular position of the cursor with respect to the center of the clock. At the alarm time, the keyboard bell will sound and the clock face will turn red. The keyboard bell will sound once each minute until the alarm is reset, either by turning off alarm mode from the Alarm menu button, or by resetting the alarm time.

NAME

closeup – zoom in on an image

SYNOPSIS

closeup filename

DESCRIPTION

closeup uses the hardware pan/zoom feature of the Personal Iris and the GT products to zoom in on a portion of a bitmap image.

Middle Mouse

Holding down the middle mouse button will expand the area inside the blue box to fill the entire window.

Left Mouse

Clicking the left mouse button make the area that is expanded smaller (effectively increasing the level of magnification).

Right Mouse

Clicking the right mouse button make the area that is expanded larger (effectively decreasing the level of magnification).

Keyboard Commands

The 'G' key on the keyboard will gouraud shade the expanded pixels, and the 'F' key will make them flat shaded once again.

FILES

/usr/demos/data/images for some sample images.

AUTHOR

Rolf van Widenfelt

HARDWARE REQUIREMENTS

At least 24 bitplanes.

NAME

cube – real-time display of famous cube puzzle

SYNOPSIS

cube

DESCRIPTION

Cube displays a moving, rotating, 3-D model of a well-known cube puzzle. As the cube changes, the viewpoint translates in and out and moves around and around the object. Hidden surfaces are removed in real time. The mouse valuators and buttons control the display.

To change the display, press the right mouse button to display a popup menu. Move the cursor until the menu option you select is highlighted and release the button. A menu option may have an arrow which indicates a *rollover menu* is available for that menu option. To bring up a rollover menu, continue holding down the right mouse button and move the cursor to the left or right of the currently highlighted menu entry. The rollover menu will overlap the first pop-up menu, and the choices on this overlapping menu will be active.

To get rid of all menus without changing the display, move the cursor clear of all menus and release the button.

Automatic cube operation

To operate the cube, pess and hold the right mouse button. A menu of operations appears. Move the cursor to the menu entry for the desired mode and release the right mouse button to select it. The *rotate* mode makes the cube alter itself while the viewpoint's altitude and azimuth change. *Translate* makes the viewpoint move back and forth. *Both* and *freeze* do as one would expect.

Any automatic motion will continue with the system unattended. This allows the program to be used as a stress test for the geometry system.

Manual cube operation

First select *Freeze*. The cube will come to rest as soon as its current internal motion is complete.

The left and middle mouse buttons allow the user to manually manipulate the cube. To rotate the surface of the cube, put the cursor on the desired surface. Initially, the left mouse button rotates a surface of the cube counterclockwise.

To reset the left mouse button to rotate surfaces clockwise, use the right mouse button to bring forth the pop-up menu. One of the choices on the pop-up menu is labeled *left mouse* with an arrow next to it. This menu entry has a rollover menu. There are two choices on this rollover menu: *clockwise* and *counter clockwise*. The left mouse button will be tied to whichever function you choose from this menu.

The middle mouse button has two modes. In its initial mode, pressing the middle mouse button and moving the mouse changes the viewing altitude and azimuth. In effect, this turns the cube around and brings new surfaces into view.

The middle mouse button can be tied to distance and field of view rather than altitude and azimuth. Once again, use the right mouse button to bring forth the pop-up menu. This time, find the menu entry marked *middle mouse* and move to the side of the highlighted entry to bring up the rollover menu. The middle menu button will be tied to the chosen function.

Kill terminates the program, as does the ESC key. *Reset* initializes the program and, incidentally, solves the cube.

AUTHORS

Herb Kuta and Kurt Akeley

NAME

curve – interactive cubic curve demonstration

SYNOPSIS

curve

DESCRIPTION

Curve demonstrates the use of cubic curves in both two and three dimensions. Using the mouse and keyboard, the user may create and manipulate different types of cubic curves, or view pre-defined curves. Various display options may be controlled through a pop-up menu, described below.

Mouse and Keyboard operations

Left Mouse Button adds a control point to the curve, or moves an existing control point. If the button is pressed over an existing control point, that control point will move with the cursor for as long as the button is held. Otherwise, a new control point is added to the curve.

Middle Mouse Button, when pressed in *Three Dimensional* mode, causes the view volume to rotate in a fashion similar to a trackball. If the button is pressed and held within the boundaries of the view volume, rotation follows the cursor about the x (left-right) and y (up-down) axes. If the button is pressed outside of the boundries of the view volume (but still within the window), rotation follows the cursor about the z-axis (clockwise or counter-clockwise). In *Two Dimensional* mode, the middle mouse button moves a control point in a similar fasion to the left mouse button.

Right Mouse Button brings up the pop-up menu, described below.

Backspace Key deletes the control point nearest to the cursor.

Escape Key will close the window and exit *Curve.*

Pop-up menu operations

Pressing and holding the right mouse button will bring up the top-level pop-up menu. See your *Owner's Guide* for more information on the use of pop-up menus.

The top-level pop-up menu has the following operations:

Curve Basis brings up a sub-menu which allows the basis of the curve to be changed between *Bezier Spline, Cardinal Spline,* and *B-Spline.*

Display Mode brings up a sub-menu which allows the mode of the display to be changed between *Two Dimensions* where the curve is drawn within the window in two dimensions, and *Three Dimensions* where the curve is drawn within a perspective view of a three dimensional view volume. In three dimensions, control points are given a random depth.

Options brings up a sub-menu of display options, described below.

Set Defaults sets all display options to their default values.

Clear Window clears the current window or view volume and deletes all control points.

Display options

Choosing *Options* from the top-level pop-up menu allows the following display options to be modified:

Turn Motion On (Off) toggles the motion mode. When motion is on, control points are given random velocities and directions, causing them to bounce around the window or view volume, changing the shape of the resulting curve.

Turn Markers On (Off) toggles the mode of the control point markers. When markers are on, consecutive numbers indicate the position of the control points describing the current curve.

Turn Hulls On (Off) toggles the mode of the convex hulls. When hulls are on, lines are draw connecting control points, indicating the convex hulls to which the current curve is interpolated or approximated.

Turn Smear On (Off) toggles the mode of smearing. When smear is on, anything that moves, the curve, markers, or the view volume, will "smear" on the display, not erasing its previous position.

Precision brings up a sub-menu which allows control over the number of segments used to create a spline, where each spline is defined over a series of four consecutive control points.

Speed brings up a sub-menu which allows control over the speed of motion. Higher numbers indicate faster motion.

Line Style brings up a sub-menu which allows control over the style of line the current curve is drawn with – *solid, dotted,* or *dashed.*

Whizbangs brings up a sub-menu which allows the current curve to be replaced by a pre-defined "whizbang." *Coil* replaces the current curve with a constant radius spiral. *Bed Spring* replaces the current curve with a varying radius spiral. *Doughnut* replaces the current curve with a toroid.

AUTHOR

Howard Look, based on the original program by Rocky Rhodes and Herb Kuta.

BUGS

The GL graphics library defines all curves as cubics, meaning that a curve spline is completely defined with four control points. Bezier curves with more than four control points will be drawn as a series of separate curves defined by groups of four consecutive control points.

When deleting a point in three dimensional viewing mode, the mouse position is projected onto the z=0 plane of the view volume. Therefore, points may not lie directly under the mouse on the screen.

NAME

demograph – graphs demographic data in 3D over time.

SYNOPSIS

demograph [-s states.ascii] datafile [datafile, ...]

DESCRIPTION

The *demograph* demo graphs actual U.S. cencus data in three dimensions over time. Two sets of data are available: Population Distribution or Per Capita Income by state.

Demograph operation

Moving the *time bar* at the bottom of the window will interactively update the date and the graph of the data according to the date. Earlier dates are to the left. The *middle mouse* rotates the graph and the *left mouse* translates the graph. The *right mouse* brings up a menu which allows changing the data set and changing the display method.

The –s option allows specifying the location of the binary file descrbing the U.S. map. If this option is not given demograph will look only in the local directory. One or more data files must be included in the command line.

The demograph data file is an ascii file format which allows the user to display his/her own data using the demograph display program. The first line of the data file contains an id number of 19971989 to identify it as a demographic data file. The second line contains a title which is displayed as the menu selection for this data. The title will get truncated to 16 characters. The third line contains three integers seperated by spaces: the first year of data, the total number of entrys per state, and the number of months between data entries. The next 48 lines contain the data for each of the 48 continental United States in alphabetical order. Data for each state is contained on one continuous line in chronological order, starting with the earliest entry. Data entries are seperated by space. Lines starting with a data files are read and the numbers of lines with errors are reported.

The demograph command has the following options:

Specifics:

* Population has 19 data points for each state from 1790 to 1970

* Income has 23 data points for each state ranging from 1948 to 1970

* The graph contains about 800 polygons or about 4000 vectors.

* Software dithered color map mode is used on the eight bitplane Personal IRIS. Flat shaded concave polygons are used on all other machines.

NOTE

It is interesting to see peaks of activity during certain time periods and speculate on the cause. Also note the obvious westward migration of population before the mid 1800s and the 1900's growth in California.

FILES

/usr/demos/bin/demograph
demograph display program

/usr/demos/data/demograph/population
Population Distribution data file

/usr/demos/data/demograph/income
Per Capita Income data file

BUGS

There should be a black line surrounding the inside frame. Sometimes this does not show up I believe due to rounding error.

AUTHOR

David Ligon

HARDWARE REQUIREMENTS

Eight bitplanes displays software dithered / double buffered color map. 24 bitplanes displays RGB.

NAME

dglnewton – a physical modeling demo running across a network

SYNOPSIS

dglnewton [**–f** model_catalog] [**–D**]

DESCRIPTION

dglnewton is a version of the *newton* demo compiled with Silicon Graphics' Distributed Graphics Library (DGL). When it is run, it will try to connect to the machine defined in the REMOTEHOST environment variable and send graphics to it across the network. The REMOTEHOST environment variable is automatically set when you start a remote login (rlogin) or remote shell (rsh) on another machine.

The DGL must be enabled for this demo to run; to enable the DGL, the files /etc/services and /usr/etc/inetd.conf must be edited on both machines and the network daemon *inetd* must be restarted. See chapter 7.2 of section 1 of the 4sight User's Guide for more information on setting up the DGL.

SEE ALSO

See the *newton* manual page for more information about this demo; see Chapter 7, Section 1 of the 4sight User's Guide for more information about the Distributed Graphics Library.

NAME

dglray – a visualized raytracer running across a network

SYNOPSIS

rsh machine /usr/demos/bin/dglray

DESCRIPTION

dglray is a version of the *flyray* demo compiled with Silicon Graphics' Distributed Graphics Library (DGL). When it is run, it will try to connect to the machine defined in the REMOTEHOST environment variable and send graphics to it across the network. The REMOTEHOST environment variable is automatically set when you start a remote login (rlogin) or remote shell (rsh) on another machine.

The DGL must be enabled for this demo to run; to enable the DGL, the files /etc/services and /usr/etc/inetd.conf must be edited on both machines and the network daemon *inetd* must be restarted. See chapter 7.2 of section 1 of the 4sight User's Guide for more information on setting up the DGL.

SEE ALSO

See the *flyray* manual page for more information about this demo; see Chapter 7, Section 1 of the 4sight User's Guide for more information about the Distributed Graphics Library.

NAME

dog – cooperative or competitive flight simulator and airshow generator

SYNOPSIS

dog [**–h**] [**–i** infile] [**–o** outfile] [**–b**] [**–I** *ifaddr*] [**–T** *ttl*]

DESCRIPTION

Dog is an enhanced version of *flight*(1D) that allows players on two or more networked IRISes to battle each other in a ''dogfight.'' Several times a second, each workstation sends status/location packets to the other machines using UDP multicast packets, and receives the other planes' packets. All known planes in the current field of view are displayed on all systems. Pilots may cooperate by attempting formation aerobatics or compete by trying to shoot each other down. The coordinates of projectiles are included in the packets, hits are detected, and scoring is maintained.

If either **–i** nor **–o** is selected, the status/location packets are read and/or written to files as described later under *Airshow Option.* Otherwise, the broadcast medium is the Ethernet. Older versions of *dog* on the IRIS 3000 series and IRIS-4D IRIX releases before 3.3 use broadcast packets. To have a dogfight with those systems, start the program with the **–b** option. (Note: using *dog* in broadcast packet mode may be harmful to other computers on the same network. Try to use the default multicast mode if possible.) If the workstations are configured to route multicast packets, players on several networks can fight each other. The **–T** option specifies the maximum number of times the multicast packets can be forwarded between networks. See the TCP/IP User's Guide for details on setting up multicast routing. For machines with multiple network interaces, the **–I** option specifies the outgoing interface by its Internet hostname or address.

Weapons

Fighters are armed with rockets, sidewinders, and cannon. The number of rockets and sidewinders available on each type of fighter are indicated on the help display. Landings replenish missiles as well as fuel. The number of missiles replenished depends on the quality of the landing (see *flight*(1D). Ammunition for the cannon is inexhaustible.

Each weapon has a different kill radius. Weapons detonate themselves when they are within their kill radius of a plane other than the one they came from. All planes within the kill radius of an exploding weapon are destroyed.

The *q* key fires a rocket. Rockets have about ten seconds of fuel and follow ballistic paths after the fuel is exhausted. They explode when they strike the ground, come within range of an aircraft (except the one they came from), or are destroyed by their owner. Rockets have the largest kill radius of all the weapons.

The *w* key fires a sidewinder. Sidewinders are like rockets but track, or steer themselves towards, other aircraft if they are "locked on". Sidewinders are locked on if they are fired while a target aircraft is in the orange tracking rectangle or if locked on with the *t* key. The *t* key identifies the target and locks a sidewinder onto the target for one second without firing the weapon. This is useful for identifying other planes as friend or foe. Sidewinders will not lock on aircraft lower than 150 feet, but once they are locked on, they will track a plane below 150 feet. Sidewinders stop tracking and follow ballistic paths when they run out of fuel. The Cessna 150 does not generate enough heat to attract sidewinders. A good pilot can usually outmaneuver a sidewinder unless his plane is traveling slowly with a heavy load. Sidewinders have a smaller kill radius than rockets.

The *e* key fires a cannon round. The cannon has a limited range—each cannon shell exists for only one second. Cannons have the smallest kill radius.

The *r* key destroys the current missile. Any aircraft in range of the explosion is destroyed. Each aircraft can have only one projectile in the air at a time. Missiles may take a long time to fall back to the ground after they have run out of fuel. It is therefore wise to destroy missiles that are out of fuel, allowing new ones to be fired. In *flight*, or in *dog* with no competition, strafing the airport can be good practice for the real thing.

Scoring

dog keeps track of victories and defeats. A pilot scores a "won" when a projectile fired by his plane destroys another aircraft. A pilot scores a "lost" when his aircraft is destroyed by a projectile or crashes.

Each pilot's score is displayed on his instrument panel. The scores of all the current players are shown to each new player when he joins the game and when he reincarnates himself after destruction.

When a player joins the game, an announcement is broadcast to all players. Messages are also broadcast whenever a player quits, destroys another plane, or is destroyed.

Airshow Option

The **–o** option will record the path of your aircraft on *outfile* rather than broadcasting it to the network.

The **–i** option replays a recorded flight. You will be in another aircraft, able to join the other recorded planes in formation or shoot at, but not really destroy them (your missile will explode, but the other plane will continue flying).

Specifying both the **–i** and **–o** options replays a recorded flight and produces an *outfile* containing your aircraft's path as well as the other planes' paths. *infile* and *outfile* cannot be the same file. Repeated use of the command can make formations of many aircraft.

SEE ALSO

flight(6D), shadow(6D)

AUTHOR

Gary Tarolli

NOTES

Various kinds of cheating are possible, e.g., temporarily selecting the night display to better see a distant opponent's exhaust. Some scrupulous pilots avoid operations not possible in real aircraft. Others use every trick possible.

The Cessna 150 and B-747 have 20mm cannon. This is inaccurate but amusing in the C-150 (the 747 is much too sluggish). Try taking off in the C-150 and flying around the runway, picking off opponents as they appear. Since the Cessna does not attract sidewinders, it has a chance to survive. If you make a mistake and take off in the 747, you are dogmeat.

BUGS

The cannon ammunition should be finite and the cannon should overheat and jam if used too often.

dog may core dump if too many planes join the game.

HARDWARE CONFIGURATION

12 bitplanes and 1.5 Megabytes of memory are required to run *flight.*

NAME

dragon – generates Mandelbrot and Julia sets

SYNOPSIS

dragon

DESCRIPTION

dragon generates Mandelbrot and Julia sets (a mathematical pattern vaguely resembling paisley). Each pixel in the window is considered a point in the complex plane. For each point the formula

```
z -> z*z - c
```

(where z and c are complex numbers) is iterated. The point is colored based on how many iterations it takes for the absolute value of z to get arbitrarily big. If after a number of iterations (idealy infinite) z doesn't get big the point is considered inside the set and colored black.

Mandelbrot sets are where the coordinate in the complex plane are used as the initial value for c and z starts as zero. Julia sets are where the coordinate is used for the initial value of z, and c is some constant. The default is to render the Mandelbrot set.

It first renders the set at a low resolution, sampling the set for every 3x3 block of pixels for up to 256 iterations, and then renders the set sampling at every pixel for up to 1024 iterations. The low resolution rendering is so the user may select an area to zoom in on without having to wait for a high resolution rendering to finish. The program renders the sets at pixel resolution so the bigger the window the higher the resolution and the longer it takes.

The *left mouse button* is used to sweep out an area to zoom in on. The *middle mouse button* is used to select the constant for a Julia set (the complex constant c). The Julia set is automatically started when a constant is selected.

Constants selected inside the Mandelbrot set generate connected Julia sets (topologically one blob) and constants selected outside generate Cantor 'dust.' The most interresting Julia sets are generated from constants on the boundary.

The *right mouse button* brings up a popup menu. Selecting the *reset* option resets the program to generate the Mandelbrot set from the original view. Selecting *save RGB image* writes an RGB image into a file called 'dragon.rgb' in the current directory. Selecting *save map image* writes a colormap image into a file called 'dragon.img' in the current directory.

Both types of images may be displayed with the *ipaste* command; however, the colormap image is displayed using the colors currently in the colormap. This is to allow the user to edit the colors used in the image with the color tools such as *cedit* and *interp*. The *mapimg* command can be used to convert the colormap image to an RGB image, thereby preserving the colors.

Selecting *colors* brings up another popup with various sets of colormaps.

SEE ALSO

showmap, cedit, interp, mapimg, savemap

BUGS

The algorithm is brute force. There are many faster ways of doing it. The *stripe* colors option is ugly. In the interest of speed, the program only checks for IO at the end of each row, so for 'deep' areas the response to the mouse may be sluggish.

AUTHOR

Thant Tessman

NAME

flight – simulate the flight of any of several aircraft

SYNOPSIS

flight [**–h**]

DESCRIPTION

Flight and its multi-player version, *dog*(6D), are interactive flight simulators. One large viewport shows the world; several smaller ones simulate instruments. The world is shown from the cockpit of an aircraft or from a control tower. The mouse and keyboard control the aircraft and its environment.

If the **–h** option is selected a "heads-up" display is used instead of the instrument panel. This kind of display is commonly used in the military. It allows for a larger view, which results in a slower update rate. Try it, you'll like it.

Starting Up

Flight provides two pages of help information. The first help page briefly describes the program. To freeze the action at any time and display the first page, type *h*. Read the first page and press any key to continue. The second page offers descriptions of five or more aircraft: one two-place trainer (Cessna 150), one heavy transport (Boeing 747), and at least three fighters.

Type *1* to select the Cessna 150. The view you see is from the cockpit of the Cessna. Type *d* to see the Cessna from the control tower. Type *x* a few times for a closer view. Type *d* to return to the cockpit and strike *s* three or four times to advance the throttle. The aircraft will start to taxi towards the runway. Type *F* twice to raise the flaps — Cessnas normally take off that way. When the plane is almost on the runway, tap the right mouse button five or six times to apply right rudder. The plane will start to turn right. The left mouse button moves the rudder one increment to the left; the center one sets the rudder to zero. Move the mouse till the cursor is centered on the bottom edge of the windshield and tap *s* until the thrust indicator shows all blue. When the airspeed indicator passes 50 knots, move the mouse smoothly toward you. The cursor should be in the upper center of the horizon indicator. When the rate-of-climb indicator shows blue, you are flying! Congratulations!

Now turn around and land.

Flight Controls

Flight is controlled by the mouse, the mouse buttons, and the keyboard. The mouse holds the primary flight controls.

Rightmouse and *leftmouse* move the rudder one increment to the right and left, respectively. *middlemouse* centers it. The rudder position is shown by a small red triangle at the lower edge of the artificial horizon. The rudder is used primarily to maneuver the aircraft on the ground. Airborne turns are made, as in real aircraft, by coordinated application of aileron and elevator.

The mouse X and Y valuators control the ailerons and elevator, emulating a control stick. Left-right motion controls roll; forward-back motion controls pitch. The stick position is indicated by a square white cursor. Both controls are at their neutral position when the cursor is centered at the bottom of the windshield. Stick position for level flight is slightly below center.

The *s* key increases the throttle setting; the *a* key decreases it. The left bar indicator shows the throttle setting as a percentage of full power. Reverse thrust is available and shown in red. Thrust goes to zero when the plane climbs through 50,000 feet and the engine flames out. It can be restored by descending and applying throttle. Thrust also goes to zero when fuel goes to zero. Fuel can be restored only by making a safe landing.

Secondary flight controls include the landing gear, flaps, and spoilers. To raise or lower the landing gear, type *l*. To increase or decrease the flaps, type *f* or *F*. To increase or decrease the spoilers, type *c* or *C*. Flap and spoiler ranges are determined by the aircraft. The Cessna has no spoilers and its gear is down and welded.

The landing gear has two functions: to protect the fuselage from the ground and to add drag. You may lower the gear to slow the plane down and make handling easier.

Flaps and gear are structurally unsound at high speeds. They fall off if you exceed approximately 400 knots while they are deployed. Missing flaps make good landings difficult. Missing gear makes a good landing impossible.

Flaps increase lift, increase drag, and decrease stall speed. Takeoffs are normally made with partial flaps; landings are made with full flaps.

Spoilers decrease lift and increase drag dramatically. They are most useful in dissipating excess altitude without increasing speed. While spoilers are deployed, it is difficult to recover from a stall.

Display Controls

Several controls allow the viewer to alter his view of the world.

The *left–arrow* and *right–arrow* keys rotate the pilot's point of view 90 degrees to the left or right respectively. The viewing angle (front, left, rear, or right) is displayed on the windshield. The keys are useful for looking around, but remember to set the view back to the front for any but the simplest flying.

The *d* key switches the viewpoint from the cockpit to the control tower or back. The control tower always looks toward the plane. The *x* key decreases the tower's field of view, effectively magnifying the aircraft. The *z* key increases the field of view. If there is doubt as to whether the view observed is from the cockpit or the tower, observe the center of the window: an orange cross marks the cockpit view.

The *n* key changes the time of day from daylight to night or back. There is an interesting city NNW of the airport.

The *T* key displays threat cones around surface to air missile (SAM) sights. In newer versions, SAM missiles are fired at your plane when you violate the airspace of the threat cone. SAM missiles are deadlier than sidewinders.

Instruments

This section describes the instruments on the panel from left to right. In the bar indicators, blue denotes a positive value and red a negative value.

The *thrust* indicator shows thrust as a percentage of full throttle. Reverse thrust is possible only on the ground and is used for braking.

The *airspeed* indicator is calibrated from 0 to 1000 knots. (100 knots is about 118 miles per hour.) Negative airspeeds can happen during such acrobatic maneuvers as hammerhead stalls. Since wind is not simulated, airspeed ≡ groundspeed. The numeric display at the bottom of the band displays the exact speed.

The *climb* indicator shows rate of climb in feet per minute. Note that the fighters (in normal operation) and the civil planes (usually while crashing) can exceed the 10,000 fpm maximum rate displayed. The numeric display at the bottom of the band displays the exact climb rate.

The *G-meter* indicates vertical acceleration. Each aircraft has maximum stress limits. If they are exceeded, the attitude indicator shows the message "G-LIMIT."

The *artificial horizon* helps orient the plane when the real horizon is not visible. The triangular indicator at the bottom edge shows the rudder position. If the maximum angle of attack is exceeded, a "WING-STALL" message is displayed and a warning bell sounds. The more severe the wing stall, the less control you have over your plane. Very severe stalls may throw your plane into a violent spin.

The *heading* meter displays a combination compass and radar screen. The compass rotates and indicates your heading. Your plane's location is always at the center of the radar screen. The radar screen shows the positions of the runway and planes that are within a few miles of your aircraft. The blue line indicates the position of the runway. In *dog*(6D), other planes are shown on the *heading* meter as red blobs if they are above you or green blobs if they are below you.

The *fuel* gauge shows remaining fuel as a percentage of a full tank. To reduce fuel consumption to zero (for tests only) type ˜. Note that this forfeits all your missiles and is considered cheating.

Landings and Crashes

A good landing is a landing on the runway, with gear down, a descent rate of less than 600 fpm, and wings level. Good landings are rewarded with scores from 0 to 100 points. Points are subtracted from a perfect score of 100 based on touchdown location, descent rate, roll, heading, and drift. For every point scored, fuel on board is increased by 1% of total capacity until your tank is full. For every ten points scored you receive a missile up to the plane's limit.

Landings with the gear up, descent rate, roll, or drift too high, but not disastrous, count as crash landings. You can keep flying, but get no more fuel nor ordnance.

Landings off the runway are ''crashed into the swamps'' landings. Landings with excessive descent rate, roll, or drift are ''EXPLODED ON IMPACT'' landings. In either case, all you can do is look at the wreckage from the tower or restart the game.

Restarts

Your plane is destroyed if it crashes, taxis too far off the runway, raises the gear while on the ground, or is shot down. After your plane is destroyed, *r*, *R*, *u*, or *U* reincarnates your plane and restarts the game at the second help page. You then choose which type of plane you want to fly. These different restart options are included to make it easier to restart in intense *dog* combat. Without them, some pilots simply hang around the runway and blast new planes as they appear.

The *r* key restarts you at the original starting location. The *R* key restarts you at the south end of the runway. The *u* key reincarnates your aircraft at a random location in the sky with some randomly low airspeed and full throttle. The *U* key is similar but starts your plane at a fixed location beyond the threat cones and mountains. From here you can practice flying towards the threats and then inbetween the cones. Be carefull!

The *I* key stops the program and stows it into an icon. Open the icon to resume the game.

Weapons

See *dog*(6D) for a complete description of weaponry.

Weight

Flight models aircraft weight accurately. Ordnance and fuel have substantial weight. As you fire weapons and burn fuel, your plane becomes lighter and more maneuverable.

GT Features

In the GT version the time of day is set according to when flight is started up. As the real time changes so does the position of the sun. The *n* key adds five minutes to the time of day. The *N* key subtracts five minutes from the time of day.

SEE ALSO

dog(6D), shadow(6D)

AUTHORS

Original version by Gary Tarolli.
Enhancements by Barry Brouillette and Gary Tarolli.
GT/GTX version by Rob Mace.

NOTES

As noted above, a "heads-up" display is available. Further improvements include 3-D mountains, several bug fixes, a P-38 fighter plane, and retracting landing gear.

BUGS

Flight and its offspring are continually being improved. Improvements may be documented in the program's help display before this document is updated.

The Cessna is too difficult to bring out of a stall.

In the GT version, the F15 and F18 planes all look like an F16.

HARDWARE CONFIGURATION

12 bitplanes and 1.5 Megabytes of memory are required to run *flight*.

NAME

flip – spin one or more objects

SYNOPSIS

flip modelname [modelname, ...]

DESCRIPTION

Flip is a replacement for the old *spin* demo. It tries to read each of the named model files first from the current directory and then, if not found, from /usr/demos/data/models. The models all come up spinning around random axes of rotation. The number of polygons being drawn per second is shown in the bottom of the window; this number is updated every 10 frames.

Using the Mouse

The left mouse button controls all of the currently selected objects' x/y translation. Holding down the button will stop an object from spinning and make it (roughly) follow the mouse's motion.

The middle mouse controls all of the currently selected objects' rotation. The interface is a 'virtual trackball', centered at the center of the window. Essentially, the interface emulates a track-ball at the center of the window, with the modification that circular motions near the edge of the window are interpreted as z-rotations. When the middle mouse button is let up, the last spin direction is maintained; this lets you 'throw' objects. It is a lot easier to use than it is to describe.

Holding down the left and middle mouse buttons together will control all selected objects' z-translation (zoom). The clipping planes are set fairly close to the objects; be careful not to lose them by zooming them too far.

The right mouse button will bring up a rather elaborate series of rollover menus. They are described below.

Lights

Sources There are 8 light sources, arranged at the corners of a cube. Four of these sources are defined to be local lights, and four are infinite lights. This sub-menu lets you turn lights on and off. A light with the '->' characters before its name is on; selecting it again will turn it off. *Flip* starts with infite white and infinite blue lights turned on.

Light Models
The two light models you can choose from are infinite viewer and local viewer. Specifying a local viewer will reduce performance substantially.

Select lights

This option allows you to change the position and orientation of the lights; the lights can move independantly of the objects in the scene. This is most impressive when some local lights are turned on and the 'Show local lights' option is turned on.

Show local lights

This option causes local lights to be represented by colored icosahedra. Note that this will seem to have no effect if no local lights are turned on.

Objects Each object will have its own entry on the main menu. The only limit to the number of objects you may specify is the command-line length limit of UNIX. The following properties of objects may be changed:

Hide Objects may be hidden, in which case they are not drawn and not counted in the calculation of how many polygons are being drawn per second.

Select Each object may be individually selected or deselected (as may the lights). Initially, the first object specified on the command line is the only one selected. Note that deselecting everything will make the demo unresponsive to the left and middle mouse buttons.

Object Materials

Each object can be displayed in several different materials; this rollover menu displays the materials available. The material currently being used is prefixed with '->'.

Transparent

This option will only show up if you are running on a GT or GTX system and uses the alpha-bitplanes to display objects transparently.

Display Object as...

An object can be displayed as either polygons or lines. In addition, on GT products the lines may be anti-aliased and sub-pixel positioned. (Note that transparent, sub-pixel positioned, anti-aliased objects are possible).

Swirl Me

In swirl mode, objects alternately break apart into their individual polygons and then come back together. This option demonstrates that objects are made of independant polygons (and not long mesh strips).

Select All
This entry will show up only if you have chosen more than one object; choosing it will select all objects (but it does not affect the lights).

Exit
Quit the program. The ESC key will also make the program exit.

NOTE

The polygons/second number may seem low; however, *flip* is doing more than drawing polygons. Clearing the window takes a significant amount of time, and so does waiting for the vertical retrace period of the display to swap the buffer it is drawing into and the buffer being displayed (it is double-buffered to get smooth animation). For example, imagine running *flip* with an object that is a single triangle; the apparent polygons/second figure would be only 60 polygons/second, since a double-buffered program is limited to displaying 60 frames/second (the refresh rate of the video display). Giving *flip* a lot of objects to display (so it spends most of its time rendering polygons) will make the polygons/second rate go up, even though response time will go down and it will seem to be running more slowly.

FILES

/usr/demos/data/models/*.bin are objects in 'spin' format used by flip.

HARDWARE REQUIREMENTS

At least 12 bitplanes and a z-buffer.

AUTHOR

Gavin Bell

NAME

flyray – a visualized raytracer

SYNOPSIS

flyray

DESCRIPTION

flyray enables one to see a raytracer at work as if it were real and not just a computer algorithm. One can look around a database as rays shoot from a pinpoint through a piece of "film" and bounce around a scene.

The program can be explained by addressing 5 distinct parts; the choice windows, viewing window, button control area, image display, and final image window. Each of these parts are now addressed along with explanations of how they interact.

choices When flyray begins one will see three windows open. The top right window presents a list of data files. The descriptions of the data files are found in the large window to the left. The list of possible files and their descriptions are found in the file /usr/demos/data/seemodel/direct. To select a data file simply touch it with a mouse click (any button). This window also presents the "Master" Quit button. Hitting this button will break out of flyray. The Esc key will always get you back to Unix. The window in the bottom right is "intentionally left blank", to be used later.

viewing After selecting a file, the viewing window will display the data base as lighted polygons (including any spheres) on the left half of the screen. In this window one can see rays being sent through the film towards the scene. Note that for the first few scan lines all the rays do not intersect any objects, and therefore just trace off to infinity. When a ray does hit an object the point of intersection with that object is computed, and the direction of the reflected ray is determined by snell's law "angle of incidence equals angle of reflection". This ray is generated and traces off to see if it intersects anything. The lines that represent the rays show some of the color of the object they hit. After a ray and its reflected rays are finished bouncing around picking up color for the pixel on the film through which the ray was traced, the next ray is sent. In this raytracer many rays are sent through each pixel on the film. This techniques allows antialiasing; if more rays are sent out to "feel" the scene, more information is gained about the color of objects behind this pixel. The colors computed by each of these rays are averaged together to form the color of the pixel.

viewing options

To spin the scene around hold down the right mouse button with the cursor in the left hand window. The mouse's X position spins the scene along an axis that goes from the eye point where rays start through the center of the film. The Y position rolls the scene end over end. Once one gets used to this interface it is useful in orienting the scene for the best views of the raytracing. If the middle mouse is depressed the mouse controls the light source position. Since the mouse is 2 dimensional one can only move the light source in a plane. The plane of motion can be selected in the button window discussed next. Holding down the left button and moving the mouse in the X direction will zoom in and out of the objects.

control

In the top right window, many buttons are found that control the raytracer. First a note on coloring. The text in a *blue* button shows the current state, and pressing it will induce a new state. A *magenta* button shows the single event that will take place if pressed. *Final Image* will load a pre-computed image of the current data base if the image file is found in /usr/demos/data/images. If it is not found, pressing this button will case the button to turn red. After pressing Final Image, wait for a few moments for the data to be read. After viewing the final image please **exit final image with** ***close image*** **button** otherwise the whole program will exit. *Image* button selects the actual pixel by pixel resolution of the bottom right image. *Full res* selects the resolution at which the image in the left window is drawn. It turns out that drawing a polygon for every pixel of the film every frame is expensive, so reducing the resolution of this film can greatly speed up the frame rate especially if there are few rays per pixel, but Image is high resolution. *feeler* toggles in the option to show the shadow feelers. These feelers are rays sent from the intersection of a ray with an object back to the light source. If this ray is obstructed, the intersection point is in shadow. In feeler on mode the rays that are not obstructed are shown reaching back to the light. *New View* resets the observer to the starting point squarely in front of the film. The button in the top right of the window toggles various "macro modes" that are useful in demonstrations. pressing this button will put the system in a state where the button is describing which activity is the most intensive: graphics, cpu or both. The

Quit button here takes one back to the data base choice windows.

Image This window displays an unantialiased (aliased) version of the data computed by the raytracer. The image can be cleared by the New Image button. Whenever the light source is moved the image is cleared and restarted. After each rendering, the resolution of computation is increased, and the higher quality image is pasted over the old. Holding down the left button will send a packet of rays corresponding to the point under the cursor in this window. Note that you can see these rays in the left hand image. The middle button does the same, but only sends one ray. If you only have 24 bit planes for color, in double buffer mode much color resolution is lost. If one holds down the right button double and single buffer modes are toggled for this window. In single buffer the image looks sharp with 24 bits of color, but all other windows are distorted. In doublebuffer all windows will work right, but the image window has only 12 bits of color resolution.

Final Image Pressing the Final Image button loads a 900x900 prerendered image of the current data.

MISC

Try situating your viewing point to some location inside the scene, looking back at the film. Now go to the bottom right window and try to "shoot" yourself with rays.

FILES

/usr/demos/data/models/*.m	data files
/usr/demos/data/models/direct	directory of data files

DIAGNOSITICS

If data files are not found errors will appear from standard error. If a final image does not exist the "final image" button will turn magenta with "no image"

AUTHOR

Benjamin Garlick, June-August 1988. The underlying voxel raytracer is by Paul Haeberli July 1983. The polyhedra database was incorporated with much help from Jim Winget.

NAME

gamcal – visually check display calibration

SYNOPSIS

/usr/sbin/gamcal

DESCRIPTION

gamcal checks to see if the gamma correction value is correct by comparing the intensity of full on/full off horizontal lines to a region of half-intensity grey. Normally the gamma of the display should be set to 2.4 using the program /usr/sbin/gamma. When the gamma of the display is set correctly, there should be little or no change in brightness in each column of rectangles in the window. In the column on the far left, the horizontal lines alternate full intensity and black. The average brightness of the top region should be 50% white. The solid areas in the left column are filled with a constant 50% grey. If the display is properly calibrated, the solid and the striped regions should have the same apparent brightness.

SEE ALSO

/usr/sbin/gamma

NAME

gamma – get or set the gamma value stored in ~/.gamma

SYNOPSIS

/usr/sbin/gamma

DESCRIPTION

The light output of any video display is controlled by the input voltage to the monitor. Most people assume that there is a linear relationship between this input voltage and the brightness of the display. If 1.0 volts on the input of the monitor generates full brightness, we would expect 0.5 volts to generate exactly half brightness. Unfortunately this is not the case. It turns out that an input voltage of 0.5 volts generates an output brightness 0.19 times full brightness.

This is because the there is a non-linear replationship between the acceleration voltage of the display's electron beam, and the output of photons by the phosphor in the display. It is easy to approximate this non-linear response by taking the input voltage to some power, like 2.4. We can write this as:

Light Brightness = (Input Voltage)^2.4

The constant 2.4 varies from monitor to monitor. This constant may be as low as 1.4 or as high as 3.0 for certain types of monitors. This is called the "gamma" of a monitor. Our standard monitors have a gamma of about 2.4.

To experiment with this concept interactively, run /usr/sbin/gamcal. This program displays a test pattern of solids and horizontal lines. In the column on the far left, the horizontal lines alternate full intensity and black. The average brightness of the top region should be 50% white. The solid areas in the left column are filled with a constant 50% grey. If the display is properly calibrated, the solid and the striped regions should have the same apparent brightness.

Our graphics systems have hardware lookup tables that allow us to compensate for the non-linear response of the monitor. Now type "gamma 1.0". This loads linear ramps into these hardware lookup tables. Notice how the solid regions displayed by gamcal are much darker than the striped regions. This is because of the non-linearities discussed above. Now type "gamma 2.4". This will load the gamma correction lookup tables in the hardware with ramps to compensate for our monitor.

Be aware that only positive arguments make sense when using gamma. A negative value does incredibly strange and wonderous things to the screen. Use of negative values is NOT recommended!

SEE ALSO

/usr/sbin/gamcal

NAME

gview – viewer for radiosity data

SYNOPSIS

gview [**options**] **filename**

DESCRIPTION

gview is a viewer for data in 'gfo' format (the format the radiosity data is stored in). The default user interface is similar to the old 'spin' program; using the '-f' option to get the fly-through interface is highly recommended. Moving the mouse left and right turns the viewpoint left and right; moving the mouse forward looks down, and pulling back looks up. In addition, the mouse buttons and keyboard also control your point of view:

LEFT MOUSEBUTTON
moves forward

MIDDLE MOUSEBUTTON
move backwards

RIGHT MOUSEBUTTON
The right mouse button has a menu on it; the 'Auto Advance' entry does nothing unless you are running in movie-loop mode (see Options, below).

'A' KEY	magnifys the view
'S' KEY	un-magnifys the view
'W' KEY	accelerates faster
'Q' KEY	accelerates slower
'X' KEY	chooses the x-axis to be the 'up' direction
'Y' KEY	chooses the y-axis to be the 'up' direction
'Z' KEY	chooses the z-axis to be the 'up' direction (the default)
'ESC' KEY	exits the program

OPTIONS

-n	Do no graphics. GView exits before doing any graphics. This is useful for checking syntax or converting files to binary.
-f	Flying user interface. Useful for radiosity environments.
-P	Occasionally print some performance statistics.
-m file	Read named materials and lighting environment from 'file'. More than one file can be included if separated by commas.

-M mat	Make 'mat' the default material. Normally 'Default' is the default material.
-l ct,mode	Movie loop mode. 'Ct' is the number of times each movie frame is shown. A 'mode' of 0 means sequence forward. A 'mode' of 1 means sequence in a zigzag pattern. "-l 1,1" is typical.
-F mode	Enable display list filter. This option converts the display list to a format suitable for a binary file. A 'mode' of 1 causes the display list to be copied as is. No other modes are implemented.
-o str	Overlay a string 'str' above the display window. A string containing embedded blanks should be quoted.
-R f.bin	Read a binary file. Any GFO files are ignored.
-W f.bin	Write a binary file. The current display list is saved.
-w x,y	Set screen window size. If only x is provided, the window will be square.
-t x,y,z	Translate the origin of the data before it is displayed.
-s factor	Scale the data before it is displayed.
-i f.rgb	Save window to an image file f.rgb when the I key is hit.
-V file	Save viewing matrix to a file when the V key is hit.
-v file	Use viewing matrix from a file instead of interactive controls.

EXAMPLES

To create a binary file from a GFO file: (no graphics will occur, just conversion)
gview -n -F 1 -W file.bin

FILES

/usr/demos/data/gview contains some radiosity file(s).

HARDWARE REQUIREMENTS

24 bitplanes and z-buffer

NAME

hist – compute and display the histogram of an image file.

SYNOPSIS

/usr/sbin/hist inimage

DESCRIPTION

hist reads an image file specified by the user, then computes and displays the histogram of the image file. The red hash-marks indicate the boundaries of the range of intensities for a given image. The black line starting in the bottom left corner and going up to the top right is the indicator of the distribution function of the histogram.

Pressing LEFTMOUSE will print the pixel value currently under the cursor.

NAME

house – 2D to 3D architecture demo

SYNOPSIS

house

DESCRIPTION

Most architectural CAD packages today are two-dimensional applications, with 3D extensions added as enhancements to the main program. *house* is an example of what future CAD packages on the Personal Iris could be like. Obviously it is not a full package, for there is no direct user interface for modifying the model; however, it does show off the computational and graphical power of the Personal Iris.

Starting out as a 2D 'blueprint' of a floor plan a typical 2D CAD package would create, it transforms the blueprints into a 3D wireframe model of the house (use the 'automatic demo mode', selected from the popup menu). Then the walls become solid, the floors join up, the roof is added and the walls are lighted using three local light sources. The automatic demo ends by doing a building walk-through, ending up in the living room. You may then use the popup menu and mouse to control the demo.

User Interface Controls

Left Mouse

Moves the view closer (cursor at top of window) and farther away.

Middle Mouse

Moview the view up and down, left and right.

Menu Selections

Grow House

Switches between the 2D and 3D displays.

Demo Mode

Selected from the start, or from 2D view, it allows the house to grow and develop without user input. If not in 2D mode, nothing happens.

Wireframe Mode

Solid Mode

These two option toggle between wireframe and solid polygons.

Combine Floors

Separate Floors

Toggles between the floors side-by-side, or on top of each other.

Show Interior Walls

Show Exterior Walls

Add Roof
Does what they say.

Finishing Touches
Adds several objects to the house: cabinets, a front porch, etc.

Grey/Brown Exterior
Changes the house's color

Lighting Controls
Up to eight different lights can be turned on and off. Two of these lights are infinite, 6 are local.

Reset to Beginning
Resets everything back to its original state.

AUTHOR

Ed J. Immenschuh

HARDWARE REQUIREMENTS

NAME

icut – save a part of the screen in an image file

SYNOPSIS

/usr/sbin/icut screen.rgb

DESCRIPTION

icut reads an area of the screen specified by the user, and saves it in an image file. To use *icut,* place the icut window someplace other than where you wish to grab. Then, with the input focus attached (i.e. the mouse is inside the *icut* window), hold down a key on the keyboard to maintain the input focus, and then move the mouse to one of the four corners of the section of the screen you wish to save. Now press LEFTMOUSE and continue holding it down while you move to the opposite corner of the area you wish to grab, and let go when you are on top of that opposite corner. Wait until the original *icut* window disappears from the screen, before using the image.

NAME

imgexp – expand the range of pixel values in an image.

SYNOPSIS

/usr/sbin/imgexp inimage outimage [min max]

DESCRIPTION

imgexp expands the range of pixel values in an image. Pixel values less than or equal to min are mapped to 0, while pixel values greater than or equal to max are mapped to 255. If min and max are not provided on the command line, then the minimum and maximum pixel values in the image are used. This can be used to manipulate the contrast of images.

SEE ALSO

hist(6D)

NAME

insect – simulates a walking, six-legged creature/robot.

SYNOPSIS

insect

DESCRIPTION

Insect displays a six legged creature/robot walking on a plane. The speed and direction of travel are controlled in real time by the mouse. Hidden surface removal uses the IRIS graphics library *backface* command. This demo is of special interest to people interested in robotics, animation, and visual simulation.

Insect/robot/creature-from-another-planet operation

Select the insect window. The insect walks in the direction of the mouse valuators whether or not any of the buttons are pressed.

Viewpoint movement operation

Holding the *CTRL key* down moves your viewpoint forward, towards the scene. Holding the *left SHIFT key* down moves your viewpoint move backward, away from the scene.

Holding the *middle mouse button* down gives you control of your orientation. Your viewpoint will pivot in the direction of the mouse valuators.

Holding down the F key (follow) moves your viewpoint, keeping the insect in the center of your screen. Think of this operation as panning a camera (your viewpoint) to keep the moving insect in view.

Pressing *ESC* or selecting *kill* exits the program.

NOTE

Things to notice: The window can be made any shape. The scene inside automatically corrects for the aspect ratio.

BUGS

Bug is another word for insect.

AUTHOR

Thant Tessman

HARDWARE REQUIREMENTS

Eight bitplanes and 2 Megabytes of memory are necessary.

NAME

interp – gamma-corrected color ramp generator

SYNOPSIS

/usr/demos/bin/interp

DESCRIPTION

Interp makes smooth color ranges. Two color chips and a Gouraud-shaded polygon showing the shade range between them are displayed. To make a color range, start windows for *showmap*, *cedit*, and *interp*. Select the color at the lower (usually darker) end of the shade range and use *cedit* to set it to the desired value. Select the color at the top of the shade range and edit it as well.

Now attach to (select) *interp*. Point to the lower color (usually on *showmap*). Press and release the left mouse button. Point to the high color and repeat. Both the extreme colors of your desired color ramp should be in the two large color chips in the *interp* window. If you should make an error when selecting the extreme colors for your ramp, just keep trying to use the left mouse button to pick the colors, until the two color chips show the proper extreme color indices.

When the extreme color indices are correctly chosen (which is reflected in the color chips), the other colors of the ramp can be interpolated. While *interp* is still selected for input, press and release the middle mouse button. This interpolates all the colors between the extreme colors.

The resultant color array will be gamma-corrected using the floating-point constant in *~/.gamma*. If no such file exists, the gamma constant is 1.0.

BUGS

This program must have input focus in order to work properly. See the window manager documentation for a discussion of input focus.

NAME

ipaint – Paint using bitmap images as brushes

SYNOPSIS

ipaint imagedirectory

DESCRIPTION

ipaint reads images from the specified image directory and then allows you to select an image with which to paint, continuously writing the image on to the screen while the middle mouse button is held down. Note that nothing will happen until an image is chosen (see the section on the program's menus below), and that the left mouse button does nothing.

MENUS

Pressing the right mouse button brings a pop-up menu with several useful options:

Mode

8-bit will write images using only 256 colors (8 bitplanes); image quality will deteriorate.

12-bit will write the image using 12 bits (or 4096) colors.

RGB will write the image using a full 24 bits of color.

Blended RGB
will use the alpha-blending function of the GT and GTX hardware to blend the image with the pixels already on the screen; holding the middle mouse button in one position will make the image written get progressively brighter. This function only works if the system has alpha bitplanes installed.

Images This entry brings up a pop-up menu with entries for each image in the specified image directory. Selecting an image for the first time will make the screen clear and load the image; several images may be loaded by selecting them from this menu, and, once loaded, may be switched between using this menu again.

Clear Screen
Will clear the screen. Note also that the screen is cleared when an image is loaded into memory the first time it is selected.

Exit Exits the program.

BUGS

Does not distinguish between image files and other file types in the image directory specified.

Clearing the screen when an image is being loaded can be annoying.

FILES

/usr/demos/data/images is where the images for SGI demos live.

HARDWARE REQUIREMENTS

GT, GTX or Personal Iris with at least 12 bitplanes. The Blended RGB function will only work on the GT or GTX machines, and will work only if the alpha plane option has been installed.

NAME

iset – set the type of an image.

SYNOPSIS

/usr/sbin/iset newtype imgfiles

DESCRIPTION

iset sets the type of an image. This determines which part of the color map ipaste uses to display the image. The four types of viewable images are NORMAL, DITHERED, SCREEN, and COLORMAP. These are the four values newtype can have (each must must be spelled in all capital letters as above).

A NORMAL image is an RGB or monochrome image.

A DITHERED image is a color image using only 8 bits to represent the original 24-bit true RGB image. This image type is obsolete on 4D machines.

A SCREEN image contains color indexes.

A COLORMAP image is used to store colormaps.

SEE ALSO

istat(6D), rle(6D), verbatim(6D)

NAME

istat – print the header information of a list of image files.

SYNOPSIS

/usr/sbin/istat imagefiles . . .

DESCRIPTION

istat prints the header information of a list of image files. x/ysize give the image's screen size in pixels; zsize is the number of channels in the image. An RGB image will typically have three channels, while a Monochrome image will use one channel. Min and max are the range of pixel intensity values in the image. Bpp describes how many bytes are stored in each channel of the image; either 1 byte or 2 bytes. Type of image can be NORMAL, DITHERED, SCREEN, or COLORMAP. Storage refers to the way the data is compressed: rle is a run-length encoded image, verb is a verbatim image which means the data is not compressed in any way.

SEE ALSO

iset(6D), icut(6D), rle(6D), verbatim(6D)

NAME

izoom – magnify or shrink an image

SYNOPSIS

/usr/sbin/izoom inimage outimage xscale yscale [-i -b -t or -q]

DESCRIPTION

izoom magnifies or shrinks an image with or without filtering. xscale and yscale are floating point scale factors. The filtering method is one pass, uses 2-d convolution, and is optimized by integer arithmetic and precomputation of filter coefficients. Normally *izoom* uses a triangle filter kernel in both x and y directions. The -i (impulse) option causes izoom to do no filtering as the image is resized. The -b (box) option causes izoom to use a box as the filter kernel. The -t (triangle) option is the default. The -q (quadratic) indicates that a quadratic function should be used as the filter kernel.

NOTE: *izoom* does not work on dithered images which are nothing more than color look-up table indices. To perform any such image processing one must first use something like *fromdi* (in the moregltools subsystem, and also can be found in */usr/people/4Dgifts/iristools/imgtools/fromdi.c*), which converts the dithered image into an RGB image.

NAME

jello – simulates nonrigid body dynamics

SYNOPSIS

jello

DESCRIPTION

jello simulates the dynamics of an elastic body (an icosohedron) in real-time. The icosohedron is made of thirteen masses, one at each vertex and one in the middle. Each mass is connected to its neighbors by springs. Formulas for the position of the masses over time (as a function of the springs, gravity, and the walls) are iterated each frame.

The jello starts suspended in the center of a cube shaped container. Holding down the *middle mouse button* and moving the mouse around reorients the container. Pressing the *left mouse button* starts gravity and drops the jello. The *right mouse button* brings up a popup menu with display options.

Pressing the ESC key exits the program.

BUGS

It is possible to make the jello explode if it hits the wall hard enough. This is because the iteration technique (newtonian) doesn't conserve energy. Reorienting the cube actually only reorients the viewpoint and gravity. This means you can't kick the jello with the side of the container. New Jello does it right.

AUTHOR

Thant Tessman

NAME

light – demonstrates real-time lighting and shadows

SYNOPSIS

light

DESCRIPTION

Light displays a cube floating above a grid. The cube is lit up and casts a shadow on the grid. The mouse valuators and buttons control the display.

To change the display, press the right mouse button to display a popup menu. Move the cursor until the menu option you select is highlighted and release the button. To get rid of the menu without changing the display, move the cursor clear of the menu and release the button.

Automatic operation

Press and hold the right mouse button. A menu appears. Move the cursor to the menu entry for the desired mode and release the right mouse button. The *cube spin* option makes the cube rotate by itself. The *scene spin* option makes the scene rotate by itself.

Any automatic motion will continue with the system unattended.

The label on the menu is the operation that will be executed if that item is selected. It is **not** showing what state the demo is in. This means that if the menu says *cube spin on* , selecting this will turn the automatic cube spin on. It **does not** mean that the spin is currently on.

Manual operation

In this demo, the controls are a little different than are found in most of the others. The controls were designed to make smooth motions.

With the automatic scene spin off, holding the left mouse button down will make the scene spin according to the position of the mouse. Moving the mouse left makes the scene spin clockwise; to the right, counterclockwise. Moving the mouse up makes the scene spin toward you; downwards, away from you. The farther you move the mouse pointer away from the center of the screen the faster it spins in the coresponding direction.

With the automatic cube spin off, holding the middle mouse button down will control the cube in a similar manner.

Holding the left and middle mouse buttons down at the same time will make the scene move closer or farther away depending on the position of the mouse. Moving the mouse up will make the scene move away, down will make it move closer.

MTV mode

This is named after "Music Television" because it is similar to a video effect commonly used in that sort of thing. In *MTV mode* the picture on the screen itself becomes an object. It can be moved around in the same way as the scene. First get good at moving the scene around. Then select *MTV mode* on the popup menu. Press the left and middle mouse buttons and move the mouse pointer above the middle of the screen to back away from the picture a little. Spin the picture by holding the left mouse button down and positioning the mouse in the direction you want the picture to rotate.

The automatic cube and scene spin still work while in *MTV mode*. Getting out of MTV mode reverts the mouse control back to what's 'inside' the picture. You can make the scene fill the whole window again by selecting the *reset view* option. The *S*, *C* and *M* keys do the same thing as selecting *scene spin*, *cube spin* and *MTV mode* on the popup menu. One last trick: when you are in *MTV mode* and you hold the left mouse button down to spin the picture, you can press the *M* key to get out of *MTV mode*. The picture will still spin at the rate you left it.

move lights/stop moving lights

There are two light sources that may be moved independently. When you select *move lights* the left and middle mouse buttons are used to move the 1st and 2nd light sources. Holding down the left mouse button and moving the mouse moves the 1st light source, likewise holding down the middle mouse button moves the 2nd light source. Selecting *stop moving lights* reverts mouse control back to the scene.

reset lights

The light sources are both in the same position when the program begins. This option returns the light sources to that initial position. This is handy if you lose track of the lights while moving them with the *move lights* option.

reset view

This option resets the view when selected. This is handy for displaying the scene normally after using *MTV mode*.

Pressing *ESC* or selecting *kill* exits the program.

AUTHOR

Thant Tessman

HARDWARE REQUIREMENTS

Sixteen bitplanes and 2 Megabytes of memory are necessary.

NAME

liquid – A faucet dripping into a multi-colored pool of liquid

SYNOPSIS

liquid [-n <nodes>][-f <friction ratio>][-a <ambient>]
[-d <diffuse>][-s <specular>][-g <gravity>]
[-t <tiling size>][-m <mass ratio>]
[-p <pelletsize>]

DESCRIPTION

liquid is a faucet hanging over a pool of liquid, lit by an arc light. As it drips into the pool, waves spread from the impact of the drops. *liquid* demonstrates highly complex rendering, hidden surface, and wave motion, all in an abstract animated interactive scene. You can alter the position of the arc lamp, position the faucet and its drips, and change the overall viewing position.

You control the positioning of objects in the scene by pressing and holding the left mouse button on the object you want to move, dragging the mouse to the new position of the object, and releasing the left mouse button. If the mouse is not over any of the movable objects, the viewing position is altered.

You can stop all the motion temporarily by choosing the **freeze/action** pop-up menu option. You can reset the pool to its original height by choosing the reset pop-up menu option.

COMMAND LINE OPTIONS

You can use the following options on the command line. You can use any of them, all of them, or none of them when you invoke the program. You can always run *liquid* with the defaults, and choose not to use any of the options.

-n <nodes> sets the ratio of the number of nodes per side. This is a measure of the subdivision for the pools' surface. The default is eight.

-f <friction> sets the dampening of the wave motion. Values range from 0.0 (no dampening) to 1.0 (full dampening). The default is very little dampening.

-a <ambient> sets the ambient coefficient of the lighting model. The default is .3, the range is from 0.0 to 1.0.

-d <diffuse> sets the diffuse coefficient of the lighting model. The default is .7, the range is from 0.0 to 1.0.

-s <specular> sets the specular coefficient of the lighting model. The default is .7, the range is from 0.0 to 1.0.

-g <gravity> sets the amount of gravity exerted on the drips as they fall. The default is -10.

-t <tiling size> sets the ratio of the floor tiling to the pool size. The default is 7.

-m <mass ratio> sets the mass ratio between a drip and the pool. The default is .125 (1/8).

-p <pelletsize> sets the ratio of the radius of drips to the size of the pool. The default is .0625 (1/16).

POP-UP MENU OPTIONS

You can use these pop-up menu options:

freeze/action– stops and restarts the motion of the liquid in the pool and the drips. Use it to set new colors and lamp or faucet positions before the scene changes.

reset– resets the position of the surface of the pool.

exit– exits the program. You must confirm your intention to exit before *liquid* exits. You can also exit by pressing the escape key on the keyboard.

AUTHOR

Eric Brechner

NAME

mag – pixel replication and magnification in a window

SYNOPSIS

/usr/sbin/mag [integer]

DESCRIPTION

Mag copies and enlarges areas of the screen. The mag program demonstrates pixel reading and writing on the IRIS. The area of the screen copied is chosen by the user with the mouse. The power of magnification is an integer value which must be entered into the command line. If no integer is specified, the default value of the power of magnification is 2.

Operator control of this demo is strictly with the mouse.

Mouse buttons	Function
left	pick area on screen under cursor

COMMENTS

Values of magnification greater than 2 significantly degrade the magnified image.

Trying to magnify the mag window causes some inexplicable results.

Sometimes, when the window manager is in double buffer mode, the area magnified appears in the wrong color. The mag program works best in single buffer mode.

BUGS

This program must have input focus in order to work properly. See the window manager documentation for a discussion of input focus.

NAME

manwsh – display a man page and then prompt for input

SYNOPSIS

manwsh [**options**] [**section**] [**title ...**]

DESCRIPTION

manwsh is a simple shell script used by the demos to display a man page in a wsh window; for example:

```
wsh -c manwsh 6 insect
```

displays the *insect* manual page in a new wsh window, and then pauses with the prompt "<ENTER> to continue" before exiting and making the wsh window go away. All arguments passed to *manwsh* are passed on to the *man* program; see its manual page for a description of all the possible options.

NAME

mousewarp, dialwarp, keywarp – set input warping parameters

SYNOPSIS

/usr/sbin/mousewarp [min mult] [off]

/usr/sbin/keywarp [min mult] [off]

/usr/sbin/dialwarp [min mult] [off]

DESCRIPTION

These three commands affect the ''feel'' of the input devices and may be used to customize their behaviour to suit individual preferences.

mousewarp sets the mouse warping parameters. *min* is the smallest raw mouse movement that will be magnified by *mult* which may be specified as a floating point number. Use "mousewarp 4 9.3" to greatly exaggerate the movement of the cursor whenever the mouse position changes by more than 4 pixels in one refresh period. The default is "mousewarp 1 1" If the single argument **off** is given, then the default values are restored. If no arguments are given, the current values are printed.

keywarp sets the keyboard auto-repeat rate. *min* is the initial time threshold before a key will begin auto-repeating. *mult* is the inter-character repeat rate while a key is being held down. Both parameters are given in units of 1/100 of a second. If the single argument **off** is given, the auto-repeat is turned off. If no arguments are given, the current values are printed.

dialwarp sets the dial warping parameters for the external dial box. The *min* and *mult* parameters have meanings similar to the parameters for *mousewarp*. If the single argument **off** is given, the default values are restored.

NAME

newave – real-time simulation of an idealized surface

SYNOPSIS

newave

DESCRIPTION

newave is a new and improved version of the wave demo. Newave displays a hexagonal grid. The vertices are masses and the line segments are are springs connecting the masses. Displacing a mass stretches the springs to other masses and send waves throughout the grid. This demo is computationally intensive and is a good demonstration of floating point power.

Holding down the right mouse button brings up a popup menu. The first item on the menu is *edit*. Selecting this puts the demo into edit mode. The grid is displayed in red with a green crosshair. The crosshairs move around with the mouse. Move the crosshair over a vertex to select it, then press and hold the left mouse button and move the mouse up or down to adjust the vertex up or down. In effect, you are pulling the mass to a new position and stretching the springs. Releasing the mouse button will leave the vertex where it is. You may move any of the vertices on the grid this way except those on the outer edge which are fixed.

After perterbing a vertex or two, hold down the right mouse button to bring up the popup menu and select *go*. This will put the grid in motion. Selecting *reverse* after a moment or two will reverse the velocities of all the masses and the grid will eventually return to the starting condition and keep going.

Selecting *reset* will return all the masses and their velocities to zero.

You may reorient the grid at any time by holding down the middle mouse button and dragging the mouse. You may also move toward and away from the grid by pressing the *up arrow* and *down arrow* keys.

There are several display modes. Selecting *display menu* brings up the display menu. *normal* is the non-depthcued wireframe mode the demo first starts in. Selecting *depthcued* displays the grid as a depthcued wireframe. Selecting *flat shaded* displays the grid as flat shaded triangles. Selecting *gouraud shaded* displays the grid as gouraud (smooth) shaded triangles. The lighting for the solid modes is recalculated each frame. Selecting *top mode* displays the grid from straight above and puts a rainbow in the depthcue ramp. This mode is purely an exercise in esthetics.

Selecting *spring menu* brings up a menu that allows you to select from weak, medium, and strong springs.

You may change the grid size by selecting *grid menu*. This brings up a pop-up that allows you to select from a small, medium, or large grid.

Pressing the ESC key or selecting *kill* and confirming on the confirm kill menu will exit the program.

HINTS AND BUGS

There is a performance penalty when using large areas of the screen. When you first start up the demo you may want to open the window a quarter to a half of the area of the screen.

The gouraud shaded stuff looks better with a bigger starting shape, e.g. pull up the center and the six around it instead of just the center. The gouraud shaded one will go really fast if the window is small and you use the small grid.

The stuff that looks best in the top view mode is stuff that is symmetrical about the center. Try pulling up all the corners. Or better yet, instead of pulling up the corners, pull up the ones on the clockwise side of the corners. Make sure they are all pulled up to the same height. Use the arrows to move the top view of the grid into the front and back clipping planes. Use the large grid and weak springs.

The solid displays are not z-buffered. If you move around to the back of them they will look wrong. If you use a large grid and switch to a smaller grid without reseting, the edge will still contain the old values. I haven't decided if this is a bug or not since it can look interesting.

Experiment around.

AUTHOR

Thant Tessman

NAME

newton – a physical modeling demo

SYNOPSIS

newton [**–f** model_catalog] [**–D**]

DESCRIPTION

Newton is a real-time simulation of an elastic body. Command-line arguments will be discussed below, after a general explanation of the program.

The body is made up of a number of *atoms* and *springs*. *Atoms* are points of mass for which the forces of gravity apply. Between some pairs of atoms, there are *springs*, which supply additional forces if their current length is different from their initial length. The specific spring equation used is not linear, but behaves close to linear in a narrow vicinity of the initial spring length.

The main window of *Newton* shows a cubic room that contains the elastic model at its center. To drop the model, press and release the *left* mouse button. Between the time you press the left mouse button and the time you release it, you get a chance to reorient the model any way you want – just move the mouse around and the model will turn in that direction. Independently, you may wish to rotate the room. This can be achieved by pressing the *middle* mouse button and moving the mouse around. To stop reorienting the room, let go of the middle mouse button.

Whenever you want to re-drop the model, hit the left mouse button. As before, you get a chance to reorient the model prior to dropping it (you drop it by releasing the left mouse button).

MENUS

As usual with the *GL* demo programs, the *right* mouse button is the menu button. Several menu selections are available:

models There are a number of different models to select from, and a different shape can be selected via the model catalog menu.

physics The simulation is controlled by several physical parameters, and they can all be changed by the user. For example, if the user wants to increase the gravity (essentially, make the model heavier), all she has to do is select ''gravity'' in the ''physics'' submenu, and a gravity slider will pop up. See the section on sliders below to find out what physical parameters are available, and how to use the sliders.

model display The model can be displayed in several fashions:

smooth surfaces Display the model as a single surface, using the lighting model to obscure the corners of the body;

flat surfaces The normal display mode for most models. Every surface is lit independently;

springs Display the internal connections between the ''atoms'' of the model;

Bermuda This is a weird display mode using the color map, try it!

NOTE: Only one of the above four is possible at any given time, so if you are tired of the *Bermuda* display mode, the way to return to a more normal display mode is to simply select *flat surfaces, smooth surfaces,* or *springs* from this submenu.

toggle translucency
The model can be made either opaque or transparent. Consecutive selections of this menu item toggle between these two possibilities.

toggle surfaces+springs
It is sometimes helpful to see both the surfaces and the springs of the model at the same time. If you have selected either *flat* or *smooth* surfaces, and you want to superimpose the springs, click this menu item. Clicking it again cancels the springs. This is particularly useful if the model is made transparent, using the previous menu item.

room display This controls the way the room itself is drawn. Selections are either *lighted walls w/ shadows,* which means the walls are lit (just like the model itself,) and the model casts shadows on them; *lighted walls w/o shadows,* which is faster (since there are less polygons that need to be drawn); and *pinball walls* , which are non-lit walls, but rather walls that light up whenever the model hits them (the color represents the amount of displacement).

spin mode on/off

As with many *GL* demos, there is a mode in which things happen "by themselves" without user intervention. Turning *spin mode* on causes the room to continually follow the mouse, rotating in the mouse direction with a velocity proportional to the distance from the mouse position to the center of the *screen* (*not* the model window).

exit

Quit *Newton*. Other ways of quitting include hitting the ESCAPE key, and selecting **quit** from the menu bar.

SLIDERS

A *slider* is a means of changing the value of some physical parameter of the system. If the slider you need is not open already, you can open it from the **physics** menu, as explained above. A slider is essentially a window that shows the lowest, highest, and current values of the corresponding physical parameter: the precise values appear in the lower left, lower right, and lower middle of the slider window. A visual interpretation appears above the numbers as a rectangle that is partitioned into a green and yellow sections, which correspond to the portion of the range below and above the current value, respectively.

Moving to any point within the graphic representation of the slider and clicking the *left* mouse button will make the value corresponding to that point become the current value of the slider. You can also slide the value by using the *middle* mouse button in a manner analogous to rotating the room (see above). The *right* mouse button brings up the slider menu, which enables you to reset the slider to its default value, or close the slider window altogether. Once closed, a slider window can be reopened from the **physics** menu.

Available sliders are:

Gravity	The magnitude of the gravity vector. It always points down (in screen space).
Spring Constant	The spring constant of the stiffest spring in the model.
Wall Stiffness	The walls of the room are like trampolines, and that is why the model bounces off of them. This parameter controls the stiffness of those trampolines. The higher the value, the harder the walls kick back. The lower the value, the soggier the walls. The latter results in the model "sinking" into the walls.

Wall Friction When the model hits a wall, it typically loses some energy due to wall friction. This parameter controls which fraction of the energy is lost. The higher the friction, the more energy gets lost. Note that with high friction the model often ''prefers'' ''jumping'' along a wall to ''sliding'' along it.

Air Dampening This parameter controls how much energy the model loses simply by moving through the air that's inside the room. When this value is high, it is as if the model is surrounded by a viscos material (such as honey) rather than air. When this value is zero, the model experiences no air resistance whatsoever.

Display Step If you think of what you see on the screen as a movie, this parameter controls which frames actually get drawn. When the display step is 5, for example, only every fifth frame (roughly) of the movie gets displayed. When the value is high, the animation is usually faster and jumpier. When the value is low, the movie is more smooth, but has a feeling of slow motion.

COMMAND LINE ARGUMENTS

The **–D** (demo-mode) option causes all the sliders to be opened (as well as the main window) in pre-defined positions on the screen. Specifiying ''–f model_catalog'' causes the program to use an alternative list of model shapes instead of the default ones. The serious user may experiment with new model shapes once she managed to decipher the obscure format of a model description file...

A model catalog is a list of model description file names. If a name is not fully-qualified, it is considered relative to the directory containing the model catalog file.

If the model_catalog is '–', it is taken to be the standard input. For example, to have *Newton* run on all the ''*.j'' files in the current working directory, you might use

```
echo *.j | newton -f -
```

HELPFUL HINTS

There are many fun things to do with this program. However, remember that certain combinations of physical parameters may cause the model to break – very much as in real life. When the model breaks, there is no need for panic: simply hit the left mouse button and you get a fresh model that you can break again ...

You can kick the model by tilting the room so the lowest point is a corner. Let the model come to a rest at that corner, and then rotate the room around a horizontal axis – simply drag the mouse either up or down rapidly with the middle mouse button pressed. The decision whether to go up or down depends on the position of the low corner: if it is more to the front (facing you) – go up, if it is facing away from you - go down. Kicking normally introduces gobs of energy into the system, and some models do not handle that very well. You may want to decrease wall stiffness or increase spring constant before kicking.

Another fun thing is to see soggy walls: just push wall stiffness to a very low value, and then drop the model. This is particularly visible if you align the room so that the bottom wall is almost flat, but still faces you a little (if a wall faces away from you – it automatically becomes transparent) prior to dropping the model. When the model comes to a rest, you may kick up the wall stiffnes (select the default wall stiffness from the slider menu), and the model will soar to the sky as fast as gravity and the air dampening enable.

Once you have mastered the user interface to this program, try to get the *chain* model to hang in free space by its two endpoints. In other words, get the room aligned so that there is an edge of the cube at the bottom, and the two walls incident on that edge form a V-shaped corner. Then drop the model. Wall friction and stiffness may help you overcome inacuracies in the room alignment. Once the model hangs there, you can gradually lower and raise the spring constant, lower and raise gravity, or kick the wall stiffness – all of which will show amusing effects on the chain.

These are but a few of the possible experiments that can be carried out using this simulation.

FILES

/usr/demos/data/newton/model_catalog the default list of model shapes
/usr/demos/data/newton/*.j model description files

DIAGNOSITICS

Self explanatory. Messages that appear on the terminal from which *Newton* was invoked indicate a problem *Newton* is experiencing in performing the simulation. Typically, this indicates an incorrect setup of either the *Newton* program or the catalog or description files.

BUGS

The bending of the walls is approximated by a crude pyramid. The *sheet* model particularly suffers from that, so you have to have relatively stiff walls to get a decent performance out of the *sheet*.

NAME

redirect – run a demo with error output directed to /dev/console

SYNOPSIS

redirect command_line

DESCRIPTION

redirect is a simple shell script used inside the *demochest* program to make demos run from its menu print error messages to the console window. It is not meant to be run interactively.

NAME

revolve – surface of revolution demonstration

SYNOPSIS

revolve

DESCRIPTION

Revolve demonstrates a technique of creating solid objects called "Surface of Revolution". This method creates an object by sweeping a two dimensional curve about a fixed axis to create a volume in three dimensions.

Two windows, the *grid window* and the *solid's window*, make up the interface to *revolve*. Press the <Esc> key while the cursor is within either window to exit the program.

Grid Window

Left Mouse Button, when clicked within the grid window, adds a control point to the piecewise linear curve at the location of the cursor. The control point may be moved with the cursor for as long as the left mouse button remains pressed. Clicking between two existing control points will insert a control point into the curve.

Middle Mouse Button, when clicked within the grid window, grabs the control point nearest to the cursor and allows it to be moved for as long as the middle mouse button remains pressed.

Left Shift Key, if pressed while the cursor is within the grid window, causes the grid to move in conjunction with the cursor.

Left Ctrl Key, if pressed while the cursor is within the grid window, causes the grid to be zoomed in or out. The degree of magnification increases with the cursor's distance from the center of the grid window.

Backspace Key, if pressed while the cursor is within the grid window, deletes the control point nearest to the cursor.

Solid's Window

Left Mouse Button, if pressed within the solid's window, translates the solid in conjunction with the motion of the cursor.

Middle Mouse Button, if pressed within the solid's window, rotates the solid using a virtual trackball interface.

Left and Middle Mouse Buttons, if pressed together within the solid's window, cause the view of the solid to be zoomed in or out in conjunction with the cursor.

AUTHOR

Thant Tessman

NAME

rle – force an image to be stored using run length encoding

SYNOPSIS

/usr/sbin/rle inimage outimage

DESCRIPTION

Sometimes images are stored with no compression in verbatim format. *rle* converts an image to be stored using run length encoding.

SEE ALSO

iset(6D), istat(6D), verbatim(6D)

NAME

rotimg – maps an image onto a surface

SYNOPSIS

rotimg <imagefile>

DESCRIPTION

rotimg takes an RGB image file as input. It maps the image onto a surface that the user can then manipulate. The user can select between puting the image onto a flat surface puting the image onto a wavy surface. The program uses triangle meshes to display the surface. Each vertex in the triangle mesh is colored based on the input image. As the image is rotated the image is displayed at a lower resolution. When the user lets go the image increases resolution.

The *left mouse button* zooms toward and away from the image. The *middle mouse button* reorients the image. The *right mouse button* brings up a pop-up menu that allows the user to select between the *flat* and *wavy* surface.

SEE ALSO

ipaste

AUTHOR

Thant Tessman

NAME

scrsave – save a part of the screen in an image file

SYNOPSIS

/usr/sbin/scrsave outimage.rgb [x1 x2 y1 y2]

DESCRIPTION

scrsave saves a portion of the screen into an image file. *scrsave* with no arguments will save the contents of the entire screen to an IRIS image file. By giving additional arguments a smaller region of the screen can be saved. *scrsave dump.rgb 0 99 0 99* saves a 100 pixel by 100 pixel square in the lower left hand corner of the screen.

SEE ALSO

icut(6D)

NAME

shadow – full-screen armchair pilot's view of the dogfight

SYNOPSIS

shadow [**–i** infile]

DESCRIPTION

Shadow allows passive observation of the *dog* environment. The *shadow* operator sees a full-screen view of the world from a selected aircraft or of a selected aircraft from the control tower.

The program asks your name and broadcasts it to the *dog* world. The initial display is a help menu. Any key starts the program. At any time during the operation of the demo, the *h* key brings back the help page.

The *d* key toggles the view from the cockpit to the control tower. The *z* and *x* keys increase and decrease the field of view in the tower view. The *arrow* keys select front, right, left, and rear cockpit views as in *flight*. The *n* key toggles between day and night views.

The *t* key advances your view to the next of several aircraft that may be in the environment.

Airshow Option

The **–i** option reads the aircraft positions from *infile*. *infile* should be created by using the **–o** option to *dog*(6D). This allows passive observation of the "airshow" recorded in *infile*.

SEE ALSO

dog(6D), flight(6D)

AUTHOR

Gary Tarolli

NAME

showmap – display the contents of the color map

SYNOPSIS

showmap

DESCRIPTION

The frame buffer of the IRIS contains values which are translated into RGB values by a color map. *showmap* displays a square filled with patches of color, one for each color index.

SEE ALSO

cedit(6D), loadmap(1G), makemap(1G), savemap(1G).

NAME

slides – slide display program

SYNOPSIS

slides filename [**filename ...**]

DESCRIPTION

slides creates animated, interactive, 2-D scenes without having to write a program. This document describes both how to use the *slides* program and how to create slides for it.

USING SLIDES

The program reads input from the given filenames, each of which is assumed to contain one slide (or screenful of information). Slides may be interactive, presenting some information and then waiting for user input before presenting some more information.

The following input is recognized:

Left Mouse Button
: Normally used to bring up additional text when the slide program is waiting for input.

Middle Mouse Button
: Brings up text quickly, without fading it in or doing anything fancy with it.

Right Mouse Button
: When all text for the current slide has been drawn, advances to the next slide, if there is one, or exits the program.

Space Bar
: Goes back to the previous slide, if there was one, or starts the first slide over again.

Escape Key
: Exits the program at any point.

MAKING SLIDES

Slide files consist of commands to be executed and text to be drawn. Command lines start with a period; all other lines are considered text lines and are drawn in the current font at the current text position. Several commands may be placed on a single line by separating them with commas; only the first needs to begin with a period. Commands may be abbreviated. Only the unique part of a command name, enough to distinguish it from other commands, need be on the command line. The available commands along with their possible arguments are described below.

All numbers are, by default, specified in pixel coordinates. A 'p', 's', or 'm' may be appended to a number to specify coordinates in terms of pixels, the size (height) of the current font, or the width of the m character in the current font. Also, '+' or '-' may be specified to indicate a relative change in a value, or '*' can be specified, which means use the default (last) value. For example, specifying '+44p' is the same as '+44', and means 44 pixels more than the last value; '3m' is the width of three characters in the current font, etc.

Text lines are drawn starting in the upper left-hand corner of the area defined by the margins (this may be changed with the **.position** command; see below). Each new line is drawn below the previous line (this may be changed with the **.continue** command; see below).

Viewing Commands

size *left right bottom top*
: change viewport and ortho2 to these paramaters, changing the part of the screen being drawn in to.

margins *left right bottom top*
: change the margins for text

Drawing Attributes

color *red green blue*
: make this color the current color for all subsequent drawing. red, green and blue should be integers in the range 0-255.

background *red green blue*
: make this color the color for the background. This command is usually used in conjunction with **clear** and **fade.**

font *index*
: make this the current font for all subsequent text drawing. Currently, there are only three fonts; font 0 is the default 12-point font, font 1 is a fancy 24 point font, and font 2 is a fancy 36 point font.

shadow *on/off number*
: draw everything following with/without a dark shadow underneath. The optional number is how far offset the shadow is; the default is 3 pixels.

underline *on/off*
: draw all subsequent text with/without underlining. Note that spaces are underlined.

fade *on/off cycles*
: fade/don't fade all subsequent text from the background to the current color. If fade is turned on, you must specify how many colors to interpolate through. '.fade off' is equivalent to '.fade on 0'.

buffer *on/off*
: turns on/off double-buffering of the display. The display is double-buffered by default.

flip *cycles*
: draws a flipping rectangle from the current position to fill up the screen.

Drawing Commands

clear clears the viewport to the background color.

pnt *x y* draws a point in the current color at (x,y).

rect *x1 y1 x2 y2*
: draws a hollow rectangle.

rectf *x1 y1 x2 y2*
: draws a filled rectangle.

circ *x y radius*
: draws a hollow circle.

circf *x y radius*
: draws a filled circle.

move *x y*
: moves the current drawing position to (x,y); see below.

draw *x y*
: draws a line from the current drawing position to (x,y) and updates the current drawing position to (x,y). Any shape may be drawn with a move followed by a series of draws.

Text Positioning

position *x y*
: specify the place to draw the next text string. x and y are measured from the bottom left-hand corner of the screen, with the margins taken into account.

slide from *x y rate*
: the next text string will slide from (x, y) at the speed of *rate* pixels per frame to the current text position (see **.position** above).

slide left/right/up/down *number*
: the next text string will slide left/right/up/down, from the edge of the screen to the current text position (see **.position** above).

center *on/off*
: draw all subsequent text with/without centering between margins.

left *on/off*
: draw all subsequent text with/without left justification (the default).

center *on/off*
: draw all subsequent text with/without right justification.

skip *number*
: move current text position down *number* lines.

continue
: used before a text to specify that the current text position shouldn't be moved down to the next line after the text is rendered. This allows several pieces of the same line to be different colors, to slide in differently, etc.

vspace *number*
: specifys how much vertical space to leave between lines of text.

Input Control

wait *on/off*
: when on, the slide waits until the left or middle mouse button is pressed before drawing the next text line. When off, no waiting occurs.

follow *on/off*
: when on, the text will follow the mouse pointer and be put on the screen only when the left or middle mouse button is pressed.

sleep *number*
: pauses the slide for *number* seconds.

EXAMPLES

Here are two example command lines:

```
.pos * 615
.color 255 255 0,shadow on,follow off,center on
```

FILES

/usr/demos/graphics/*.info - several examples of simple slides.

BUGS

Should use the font manager routines.

Constantly redraws the slide, eating up CPU time.

Should be able to resize its window.

AUTHOR

Eric Brechner

HARDWARE REQUIREMENTS

At least 12 bitplanes

NAME

snapshot – save a portion of the screen in an image file

SYNOPSIS

/usr/sbin/snapshot [-b]

DESCRIPTION

snapshot reads an area of the screen specified by the user, and saves it in an image file. To use *snapshot,* place the snapshot button window someplace other than where you wish to grab. Then, with the input focus attached (i.e. the mouse is inside the *snapshot* window), hold down a key on the keyboard to maintain the input focus, and move the mouse to one of the four corners of the section of the screen you wish to save. Now press **left** mouse and continue holding it down while you stretch out a red rubberband to the opposite corner of the area of interest. To tell *snapshot* to make the image file, go back to the *snapshot* window, press the **right** mouse and choose one of the two "Save" menu items. You can repeat this sequence in various ways until such time as you wish to exit. At this point, you can choose one of the two exit menu items with the **right** mouse. To move the *snapshot* window itself, press the **middle** mouse button.

Leftmouse functionality

The **left** mouse button stretches, reshapes, moves or starts an entirely new rubberband for you. The cursor is the constant visual indicator of what will happen if you press **left** mouse. As long as your input focus is directed to *snapshot* you will see one of 4 different cursor types depending on the location of the mouse:

camera cursor – will appear when you are on top of any area of the console screen other than on the sides or inside of the rubberband area of interest.

corner cursor – will appear when you are in the immediate vicinity of one of the 4 corners of the currently placed rubberband.

horizontal/vertical cursor – will appear when you are in the immediate vicinity of one of the 4 sides of the currently placed rubberband.

move cursor – will appear when you are fully inside the rubberband area.

When your cursor is anywhere other than on top of the *snapshot* window, whichever of the four cursors you see will tell you what will happen at that point if you press the **left** mouse button: if you see the **camera** cursor this means that by pressing the **left** mouse, you will start creating a new rubberband that you can stretch out in any direction which will stop when you let go of the mouse button; when you see either the **horizontal, vertical**, or **corner** cursors this means that pressing **left** mouse at this time will enable you to stretch the corner or side of interest and continue doing so until you release the mouse button; when the **move** cursor is visible (while inside of the rubberband), pressing **left** mouse at this point enables you to move the entire rubberband in its current shape and size until you let go of the mouse. When you see the **move** cursor, you may also press **middle** mouse to move the rubberband.

To pop the *snapshot* button window, press down the **left** mouse button while your cursor is on top of the window, and release it without moving more than one pixel in any direction.

Pop-up Menu options

Snapshot uses the gl command **fullscrn()** which has some "humorous" side effects. One of them is that unless the cursor is on top of the actual window for the graphics program (in this case, the *snapshot* button window), pressing **right** mouse will NOT bring up that program's menu. Thus, to access the pop-up menu options, you must always bring the cursor back on top of the *snapshot* button window before pressing **right** mouse to access *snapshot's* pop-menu.

The pop-up menu currently has five items defined:

- The first item reads **Save scrn as snap.rgb** if you have just started up *snapshot* and have not yet swept out a rubberband. This will create an image file of the entire console screen (notice that at this point there is a red rubberband that encloses the entire console screen). Or else it will read **Save as snap.rgb** indicating that a rubberband area of interest currently exists.
- The second item--**New file name**--will throw up a squat rudimentary textport prompting you to input a new output image file name. If, after having called up the textport, you decide you don't want to change the output image file name, simply pressing carriage with an empty string will exit the textport and not change the filename.
- The third item--**Ipaste snap.rgb**--allows you to paste up the image you have most recently made. Notice that after you have swept out some sub-section of the screen with the red rubberband, but before you have yet selected **Save as snap.rgb**, the **Ipaste** entry shows up as a grey color instead of the solid black of the other menu items. This is because you have not yet created the actual image file--hence there is nothing for ipaste to lock on to out in the IRIS universe. Once you have chosen **Save as snap.rgb**, then when you pop-up the menu again, you will see that the **Ipaste** menu item is now solid black indicating that ipaste now has a fix on the currently saved image file you have created. The same thing will happen after you have selected **New file name** but before you save an image into it.

- The fourth item--**Save and Exit**--will save whatever you currently have selected, and then exit the program.
- The fifth item--**Exit**--will simply exit the program without saving anything that may be currently defined to be snapshoted.

Notes

When you have selected the **Save ... as ...** pop-up menu item and *snapshot* is busy reading pixels, the cursor will change to an hourglass until this process is finished. Another visual cue (in case you move the cursor elsewhere and let go of the input focus) is that the word "Snapshot" that is written on top of the *snapshot* button window--which is normally WHITE--turns to RED for the duration of the pixel reading/image file building sequence. It reverts to WHITE when the image file is completed.

The **-b** option includes a bell-ringing audio cue which will then ringbell with a short duration upon completion of every **Save** operation. The text string "snapshot" which appears in the *snapshot* button window will always turn RED when an image file is being created, and return to WHITE when finished, but the **-b** ring-the-bell option was included for those wishing to be more forcefully appraised that *snapshot* is ready for more input action.

Regarding what is actually saved into your image file, the pixels that are underneath the red rubberband are NOT grabbed by snapshot. This means that where specific pixel boundaries are critical, you must be sure that what you want to make into an image file is exactly inside the red rubberband--but not underneath these red border lines. The one exception to this is when the program is first invoked. As mentioned above, *snapshot* starts up with the default red rubberband set to the full console screen. In this case, if you select **Save scrn as snap.rgb**, the red rubberband will first disappear, then an image file of size XMAXSCREEN by YMAXSCREEN will be created, and finally the red rubberband will reappear.

BUGS

snapshot makes use of the fullscrn() GL command which, as the Reference Manual warns, must be used "with caution or a sense of humor." In this case, caution is advised: when wishing to access the pop-up menu, not only must your cursor be moved back on top of the *snapshot* button window, but to work as intended, you must release whichever key on the keyboard you have been holding down to maintain the input focus while the cursor has been outside of this button window. Not releasing said keyboard button will produce "humor[ous]" results when playing with the pop-up menu.

snapshot is not yet smart enough to make sure there is enough free space on the disk partition from where *snapshot* itself was originally executed, before it blindly goes off and attempts to allocate enough memory to build an image file of the area you specify. Hence, if you find that an image that you paste up on the screen looks "funny", run DF(1) to first confirm that the disk partition that *snapshot* is running on has not had all of its "avail" space used up.

NAME

snoop – magnify and report on the screen under the mouse pointer

SYNOPSIS

snoop

DESCRIPTION

Initially, *snoop* follows the mouse cursor, magnifying a part of the screen under the cursor and reporting pixel values for the point under the cursor. Snoop operates in several modes; these are controlled from the keyboard, the mouse, and a pop-up menu.

LEFT MOUSEBUTTON
decides if snoop will read from the front or back buffers

MIDDLE MOUSEBUTTON
controls how snoop interprets the data it finds-- either RGB or colorindex

RIGHT MOUSEBUTTON
brings up a pop-up menu with several entries, many of which have keyboard equivalents:

Arrow Equivalent to pressing the 'A' key on the keyboard, in 'arrow' mode the arrow-keys on the keyboard control what part of the screen is being snooped, not the position of the mouse pointer.

Backbuffer
Equivalent to the 'B' key, will make displaying the back buffer the default (pressing the left mousebutton down will display the front buffer).

Cmode Equivalent to the 'C' key, will make displaying data as colorindices the default.

Gouraud
Equivalent to the 'G' key, will toggle gouraud/flat shading of pixels

Snoop mode controls whether or not pixel values are displayed

Zoom Equivalent to the '0' through '9' keys, controls the level of magnification.

Stop Will suspend snooping, which can also be accomplished via 'CTRL-S' (suspend) and 'CTRL-Q' (restart).

Exit exits snoop, as does the ESC key.

HARDWARE REQUIREMENTS

Runs on a GT, GTX or Personal Iris.

NAME

solidview – display the results of a finite element analysis program

SYNOPSIS

solidview [**options**] **model** [**models ...**]

DESCRIPTION

Solidview takes data calculated by a finite element analysis program and allows the user to interact with it. In its most basic form, solidview displays polygonal data and allows you to orient a cutting plane through the objects. The appearance of the models will vary depending on which 4D machine *solidview* is running on; on GT systems, the area outside the cutting plane will be semi-transparent; on systems that cannot do alpha-blending, the transparent area will be rendered in wireframe.

Data may also have additional information associated with it; this data usually represents stresses inside the object and is calculated by a finite element analysis program, although the values can represent other paramaters (i.e. temperature, turbulence, etc). This data is represented by different colors in the transparent half of the object.

Both the object itself and the stresses associated with it can be animated; for example, you can observe how the stresses on a piston change as it goes through the combustion cycle. Note that all of this information is already stored in the data file, and *solidview* merely interpolates between the precomupted 'frames' of data.

Finally, solidview can communicate with a separate analysis program through shared memory, displaying the results as they are calculated. If the environment variable **SOLIDVIEW_ANALYSIS** is set to the name of an analysis program, solidview will try to start it up and communicate with it if analysis is turned on (see the description of the analysis menu entry below).

Interface

Solidview is controlled by using the mouse, keyboard, and popup menus. Note that the popup menus are designed so that it is easy to choose several options at a time at any given submenu level; the menus stay up until you choose their gray title bar. Here is a complete description of the interface.

Left Mouse

This button moves the objects closer or farther away when the button is held down and the mouse is moved toward the top or bottom of the window.

Middle Mouse

This button rotates the object based on the mouse's absolute position on the screen.

Left+Middle Mouse
Holding both buttons down reorients the cutting plane based on the mouse's absolute position on the screen. (If iso-contours are enabled, these keys may control the iso-contours; see 'I' command below).

Right Mouse
Brings up popup menus.

Keyboard Commands

Space Bar
Will pause/unpause any animation or auto-rotation of the cutting plane and scene. Mouse and keyboard input is processed even when *solidview* is paused.

'+/-' keys
When animating, the + and - keys will step forward/backward a frame.

'S' key When run with the -iso iso-contour command line option, the 'S' key will enable the calculation and display of iso-contours.

'I' key When iso-contours are enabled, the 'I' key will make the left+middle mouse buttons control the position of the iso-contour surface. Positioning the mouse to the far left of the window corresponds to low values of stress, the right side to high values. **'XYZR' keys** These keys control cutting-plane motion (with the left and middle mouse buttons held down). 'R', the default, will allow you to rotate the cutting plane. 'X', 'Y', and 'Z' allow you to translate the cutting plane along its YZ, XZ, and XY axes, respectively. **'D' key** This key will dump the current values for the scene and cutting plane rotations to standard output in the form of ATTRIBUTE commands that can be put inside model files to control the default orientations and spin.

Menus

Parts This menu changes based on what parts have been added to the display. The currently selected part will have arrows around its name; the To choose a different object, just select that menu entry. The 'Add Model' option lets you add all of the parts in a model to the display. Note that this will be the only entry on the menu if no parts have been added. Also note that a part may be added more than once (although it doesn't make a lot of sense to do so). 'Delete' removes the part from the display; you may add it again using 'Add Model'. 'Toggle' will toggle the part on and off; if it is off, it is not displayed. 'Modify' will bring up another set of menus that allows you to change how the part is displayed. The first eight entries let you choose the paramater you want to change (it will be surrounded by arrows); the 'Display' sub-menu changes how (or if) that part is displayed. Usually, only the front, back and cut surfaces of the part are displayed. Here are descriptions of the surfaces that can be displayed and how they can be displayed.

All Polygons
: Allows you to display all of the polygons (both internal and external) in the object (see the 'Display' options below).

Outer Surface
: Controls the display of the polygons on the outer surface of the model.

Front Surface
: Controls the display of the polygons in front of the clipping plane. They are usually displayed as alpha-blended stress values (lines on machines that can't alpha-blend).

Cut Surface
: Controls the display of the polygons formed by the intersection of the model and the cutting plane. Usually they are displayed as polygonal stress values.

Back Surface
: Controls the display of the polygons in back of the clipping plane. Usually they are displayed as lighted polygons using the 'silver' material.

Iso-Contour Surface
: If solidview is started with the '-iso' command line option, then it can comput iso-contour surfaces inside the data. An iso-contour surface is a surface inside the model where all stress values are the same. You must first turn on iso-contouring by pressing the 's' key on the keyboard. Pressing the 'i' key will make the left+middle mouse buttons control which contour to display instead of reorienting the clipping plane. Turning off other the display of other parts, turning on this option, and then animating the stresses can yield a very nice display.

Front Surface of Iso-Contour

Back Surface of Iso-Contour
: These two options don't do anything in the current version of *solidview*.

Display The options in this sub-menu allow you to change the display of whatever paramater is chosen above.

Polygons
Display as solid polygons. The color of the polygons depends on the Stresses, Materials, User Materials, and Alpha options (see below).

Lines Display in wireframe. The color of the lines depends on the Stresses, Materials, User Materials, and Alpha options (see below).

Hidden Lines
This option is currently broken, but will be fixed.

Stresses The colors of the polygons or lines displayed will be determined by the stress values in the model at that point.

Materials
The colors of the polygons or lines will depend on the surface normal at each point and the material chosen. Note that currently iso-contour surfaces have no surface normals, so specifying a material for them doesn't work properly.

User Materials
If you use the -umats command-line option, this entry will allow you to select a user-defined material. See the file **/usr/demos/data/solidview/body.mat** for an example.

On/Off Turns display of this part of the model completely on or off.

Alpha Allows you to specify the alpha-blending component of the part. An alpha of 0 will make the part totally transparent, an alpha of 255 will make it totally opaque.

Motion These options allow you to control how things move.

Spin Scene
If on, the entire scene will rotate by itself. This option is turned on by default. The space bar will pause the display until it is hit again.

Spin Plane

on, the cutting plane will rotate by itself. This option is tured off by default. The space bar will pause the display until it is hit again.

Animation

A *solidview* file may contain several 'frames' of data, showing a model in various positions with various stresses associated with it. *solidview* can interpolate between these frames, generating real-time animated sequences.

Displacement

Allows you to animate the movement of models. Either the entire model may move (in the case of the piston model, for example), or part of the model may stretch/bend/twist (in the case of the beam models, for example).

Stress Allows you to animate the stress values associated with a model as it moves.

Type *Solidview* will interpolate the data in two different ways; using a sine wave, which gives a smooth, cyclic effect, or as a simple linear ramp, running from the beginning of the cycle to the end and then abruptly starting over again.

Range Allows you to control the range of time values over which the animation takes place.

Cycles Allows you to control how finely *solidview* interpolates the intermediate stresses and displacements; a large value gives you very smooth motion, but is correspondingly slower.

Reset Resets animation values back to their default values.

Analysis

Attempts to start up the program specified in the 'SOLIDVIEW_ANALYSIS' environment variable and communicate with it through shared memory to generate displacement and stress values in real time. This option is likely to change or disappear with the next version of *solidview;* more general analysis is being integrated into the code. Documentation on the shared library interface will be written when the code is stable.

Reset Resets the view and turns the 'Scene Spin' and 'Plane Spin' options off.

Stress This entry allows you to determine which stresses are displayed; sig1, sig2, and sig3 correspond to the element of stress in the x, y, and z directions, while SI and SE are the elements of stress in the two vector coordinates. Real Time Stress will be supported in the next version of solidview.

Light Color
Lets you change the color of the light source.

Light Type
Lets you change the type of light (inifinite or local).

Other A few miscellaneous, useful options:

Save Image
Saves a RGB image file containing the contents of *solidview's* window into the model files' directory. Note that you must have write permission in that directory for this option to work. The filename used will be the name of the last model added to the display, with a number and the extension '.rgb' added to the name (i.e. "engine.fea.1.rgb"). Image files may be displayed using the *ipaste(1)* program.

Statistics
If on, will write lots of information about the calculations it is performing to the window it was run from.

Verbose If on, will write out information when objects are loaded and during other lengthy activities. Verbose is on by default.

Exit Makes *solidview* go away cleanly. It is a good idea to use this menu entry, since if you just kill *solidview* any programs it might have started up (like *thermal)* will hang around, eating up CPU time.

Options

Most command line options may be preceded by 'no' to turn them off; some options take arguments, which should be separated from the option by a space or tab character.

-[no]edges
Automatically detect sharp edges. If two polygons that share an edge have greatly differing face normals (controled by the edge tolerance paramater; see below), *solidview* will create separate vertex normals for each polygon to preserve the sharp edge. Off by

default.

-edge_tol x
: Defines the edge tolerance paramater. 0.1 by default.

-[no]iso Allows iso-contour generation. Off by default.

-iso_mem x
: Force larger iso-contour memory. x is the number of iso-contour polygons allocated per polygon in the model, and is 1 by default.

-[no]verbose
: Inform the user of progress by writing messages to stderr. On by default.

-[no]stats
: Print statistics about the program as it does its calculations. Off by default.

-[no]1:1
: Keep aspect ratio 1 to 1 when starting. This only applies to initial size of the window; the window may later be resized to a different aspect ratio. Off by default.

-umats file
: Use the user-defined material definitions contained in named file. Normally, no user-defined materials are available.

-fovy x Define the field-of-view, in degrees. Defaults to 60.0 degrees.

-near x Define near clipping plane. Defaults to 0.1.

-far x Define far clipping plane. Defaults to 5.0.

-[no]prefpos
: Start window in a pre-defined position. Off by default.

-xorg n Define x-origin used by -prefpos. Default is 100.

-yorg n Define y-origin used by -prefpos. Default is 100.

-xdim n
: Define width used by -prefpos. Default is 800.

-ydim n
: Define height used by -prefpos. Default is 800.

-machine str
: Force machine type to str. Defaults to the string returned by the *gversion(3)* call. This affects the default setting of -alpha, -czclear and -quad (see below). See *gversion(3)* for valid machine names.

-[no]alpha
Start in alpha-blending mode. On by default for GT-type machines, off for G's and Personal Irises.

-[no]czclear
Allow czclear usage. Czclear is a command that speeds up graphics by clearing both the color bitplanes and the z-buffer on certain machines. On by default for machines that benefit from it.

-[no]quad
Force quad-word alignment. Data aligned on quad-word boundaries is sent to the graphics hardware much faster, improving performance. On by default.

FILE FORMATS

Currently, *solidview* accepts three kinds of input formats. One of the formats is obtained from the results of the ANSYS finite element program. The other format is derived from the results of FEAP (Finite Element Analysis Program) and the third format is polygonal data. The following sections explain the syntax of these three formats. The text following the ';' is a line by line explanation of the syntax; it is not the part of the input file. The words in lower case are variables. Solidview files should be named 'filename.fea', although this naming convention is not enforced.

FORMAT I (Ansys)

Following is the content of the input file obtained from ANSYS:

```
PART: partname       ; the name of the part. A file can have multiple parts
                     ; partname cannot have embedded blanks
FEM: ANSYS           ; indicates that the part is in ANSYS file format
NODES 6              ; indicates that a list of nodes follows
nn                   ; number of nodes
1  x y z u v w       ; x y z are x y and z coordinates (real)
2  x y z u v w       ; u v w are currently ignored but need to be present
       .

       .
nn x y z u v w
ELEMENTS 12          ; indicates that a list of elements follow
ne                   ; number of elements (must be 8 noded brick elements)
1 mat rel type n1 n2 ... n8 ;  set mat = rel = type = 1
2 mat rel type n1 n2 ... n8 ;  n1 to n8 defines the element
       .

       .
ne mat rel type n1 n2 ... n8 ;
STRESSES 5           ; indicates that a list of stresses follow per node
ns                   ; ns = nn
1 s1 s2 s3 s4 s5 s6  ;
```

```
2
      .
      .
ns
DISPLACEMENTS 3     ; list of nodal displacements follow
nd                  ; number of displacements entries = nx
1 d1 d2 d3          ; x y and z displacement (real) of a node
2
      .
      .
nd d1 d2 d3
PART: part2         ; if a file has more than one part ...
      .
      .
```

FORMAT II (FEAP)

Following is the content of the input file obtained from FEAP:

```
PART: partname      ; part name (no embedded blanks allowed)
FEM: FEAP           ; indicates that the part is in FEAP format
PC-FEAP: Three Dim.. ; title card
nn ne nm nsd ndn nne ; # of nodes, # of elements, # of materials,
                    ; # of spatial degree of freedom (3), # of degree of
                    ; freedom per node, # of nodes per element (8).
x y z               ; x y and z coordinates of nn nodes.
      .
      .
x y z               ; last node (nn)
n1 n2 n3 n4 n5 n6 n7 n8 m1 ; n1 to n8 are nodes of brick element, m1 is mtl.
      .
      .
n1 n2 n3 n4 n5 n6 n7 n8 m1 ; last element (ne)
b1 b2 b3            ; boundary condition codes per node per DOF. (integers)
      .             ; eg. if ndn = 3 we will have 3 entries per line
      .             ; -1 means fixed DOF, 0 means free
      .
b1 b2 b3            ; last node (nn)
f1 f2 f3            ; specified forces and displacements code (real)
      .             ; -1 means displacement, 0 means force
      .
f1 f2 f3            ; for last node (nn)
t1                  ; nodal temperatures
      .
      .
tnn                 ; for last node (nn)
```

```
m1 m2 m3 m4 m5 m6 m7 ; material dof map (integers) per material
mv1 mv2 mv3 ... mv18 ; material values (real) per material
Nodal Stess = ...   ; Stress title
ns nn               ; # of stresses and # of nodes
s1 s2 s3 ... sns    ; stresses per node
    .
    .
s1 s2 s3 ... sns    ; last node (nn)
Displacement Time .. ; title for displacement
nd nn               ; # of displacement and # of nodes
d1 d2 ... dnd       ; displacement per node
    .
    .
d1 d2 ... dnd       ; last node (nn)
```

FORMAT III (Polygonal Data)

Following is the content of the input file which has polygonal data :

```
PART: partname      ; name of the part (no embedded blanks allowed)
ATTRIBUTE: MATERIALS m1 m2 m3 m4 m5 ; these are materials for diff. polygons
                    ; 0 is default, 101,102 are user defined mtls.
                    ; m1 if for all polygons
                    ; m2 is for outer polygons
                    ; m3 is for polygons behind the cutting plane
                    ; m4 is for polygons on the cutting plane
                    ; m5 is for polygons in front of the cutting plane
ATTRIBUTE: SCENE_ORIENTATION    ; xyz angle, representing the axis
                                ;  to rotate around, and amount to
                                ;  rotate
ATTRIBUTE: PLANE_ORIENTATION    ; xyz angle
ATTRIBUTE: SCENE_ROTATION    ; xyz angle
ATTRIBUTE: PLANE_ROTATION    ; xyz angle
FEM: POLYGON          ; indicates that the part is in polygonal data
Polygons generated by ... ; title card
nn              ; number of nodes
x y z           ; x y and z coordinates for nodes
    .
    .
x y z           ; for last node (nn)
np              ; number of polygons
m nv n1 n2 n3 ... nnv; material type, # of vertex per poly, vertex list
    .
    .
m nv n1 n2 n3 ... nnv; for last polygon (np)
Stresses          ; title card
```

```
ns nn            ; number of stresses (<= 6) and number of nodes
s1 ... sns       ;
      .
      .
s1 ... sns       ; for last node (nn)
Displacements        ; title card
nd nn            ; number of displacement and number of nodes
d1 ... ndn        ; displacement per node
      .
      .
d1 ... ndn        ; for last node (nn)
```

FILES

/usr/demos/data/solidview/ contains sample data files. These are ascii files; see them for examples of the formats accepted by *solidview*.

BUGS

The menu structure should be replaced with a panel interface.

Front/Back surface of iso-contour option doesn't work.

Hidden line option doesn't work.

The program exits if it is unsuccessful at opening the image file for the 'Save Image' option.

AUTHOR

Jim Winget

NAME

swap – demonstrates swapping buffers to display smooth animation.

SYNOPSIS

/usr/people/tutorial/c.graphics/online/swap

DESCRIPTION

swap simulates drawing an object to both front and back buffers, and also swapping those buffers. You can experiment with the *frontbuffer, backbuffer* and *swapbuffers* Graphics Library routines. There are four main sections of the window: a help area, an onscreen menu, a representation of the front buffer, and the contents of both front and back buffers.

Display Windows

swap has four different windows. The information window displays instructions for using this program. The console window contains three menus. Use the top menu to clear or draw to all enabled buffers, to swap the contents of the front and back buffers, and to exit from the program. Use the middle menu to enable and disable the front buffer for drawing (by default, the front buffer is disabled). Use the third menu to enable and disable the back buffer (by default, drawing takes place in the back buffer). Always use the left mouse button to select items from these menus. The screen view window is actually an enlargement of the contents of the front buffer, that is, the buffer that contains the image that you see. The buffers window shows the contents of both front and back buffers. When you enable a buffer, it is highlighted with a contrasting color.

WINDOW MANAGER

swap runs only in the window manager.

HARDWARE REQUIREMENTS

Your IRIS needs 12 bitplanes of image memory to run this program.

AUTHOR

Mason Woo

NAME

verbatim – force an image to be stored without run length encoding

SYNOPSIS

/usr/sbin/verbatim inimage outimage

DESCRIPTION

verbatim converts an image to be stored with no compression. Usually images are stored using run length encoding in rle format.

SEE ALSO

iset(6D), istat(6D), rle(6D)

NAME

vortex – display computation fluid dynamics calculations

SYNOPSIS

vortex [**-n #particles**] [**-h**]

DESCRIPTION

vortex communicates with a CFD analysis task through shared memory, and displays the results of the CFD calculation by displaying particles moving through the flow field. The simulation is of a jet hitting a flat plate in the presence of a cross-wind. The color of each particle represents the pressure at that point.

When vortex is run, two windows are opened; first, the main viewing window, and then a smaller window containing two slider bars. The program then waits for the analysis task to get started; it will start displaying particles as soon as the analysis has begun.

Using the Mouse

Left Mouse

moves the particle source inside the volume. When pressed, 3-dimensional cross-hairs appear inside the volume. You can move along any of the three axes by moving the mouse in the direction of the axis; movement between two axes is interpreted as movement in the plane containing those axes. You may constrain the movement to just one axis by holding the shift key before you press the left mouse button.

Middle Mouse

rotates the entire scene. You can spin the scene by 'throwing' it; depress the middle mouse button, then let go of it while moving the mouse-- the object will continue to rotate in the direction you threw it.

Right Mouse

brings up a popup menu that lets you control various aspects of the display.

Sliders The two slider can be adjusted with the left mouse button. They control the size of the movable particle source, and the distance each particle travels each time it is updated (this controls how fast the particles seem to flow).

Menus

Cross-flow particles On/Off

If on, particles will be created at the wall where the cross-flow originates. Off by default.

Jet-flow particles On/Off
If on, particles will be created at the nozzle of the jet. On by default.

See particle source On/Off
On by default, will represent the particle source as a shaded icosahedron.

See flow field On/Off
This option will display the entire 20 by 20 by 25 flow field as vectors. This makes a very crowded display.

Particles On/Off
Turning this off will make all of the particles disappear. Useful only if you want to see just the flow field.

Screen Clear On/Off
This option gives cheap particle traces, by not clearing the display when displaying a new set of particle positions. The display must be still (not spinning) for this option to work.

Exit Program
Exits the program. You can also exit by pressing the ESC key, or choosing 'Quit' from the window border's popup menu.

/usr/demos/data/vortex/*	input files
/usr/demos/bin/runcfd	runs vortex and the analysis task
/usr/demos/bin/earexe	the analysis task

NAME

wave – real-time simulation of the surface of an idealized waterbed

SYNOPSIS

/usr/demos/wave

DESCRIPTION

Wave displays a grid where the vertices are masses and the line segments are springs. Each segment (spring) obeys Hooke's law, F = -kx, where x is the displacement of the spring, and k is a constant. A perturbation of a mass causes a displacement of the springs which sends waves throughout the grid. The mouse valuators and buttons control the display. This demo is computationally **intensive.**

To change the display, press the right mouse button to display a popup menu. Move the cursor until the menu option you select is highlighted and release the button. A menu option may have an arrow which indicates a **rollover menu** is available for that menu option. To bring up a rollover menu, continue displaying the first popup menu and pressing the right mouse button and move the cursor to the left or right of the currently highlighted menu entry. The rollover menu will overlap the first pop-up menu, and the choices on this overlapping menu will be active.

To get rid of all menus without changing the display, move the cursor clear of all menus and release the button.

Operation

Select the wave window for input. Press and hold the right mouse button. A menu of operations appears. Move the cursor to the menu entry for the desired mode and release the right mouse button to select it.

edit--stretching the springs

With the popup menu select *edit.* The grid will turn red except for blue crosshairs on the surface of the grid. The crosshairs move around with the mouse. Choose a point on the grid and press the left mouse button. Holding the left mouse button down and moving the mouse up or down will move that point on the grid up or down. In effect, you are pulling the mass to a new position and stretching the springs. Releasing the mouse button will leave the point where it is. You may move any of the points on the grid this way except those on the outer edge which are fixed.

wave motion

To start the waves select *go*. You may select *reverse* at any time to reverse the velocities of the masses. The grid will eventually return to the starting condition and keep going. Selecting *reset* will return all the masses and their velocities to zero. The middle mouse button controls the orientation of the grid. *Kill* or *ESC* terminates the program.

Motion of the grid continues with the system unattended.

harmonics

The menu entry labeled *harmonics* has a corresponding rollover menu. The entries on the rollover menu specify inital conditions for the grid. The two numbers in each menu entry specify the number of antinodes (humps) in the x and y directions of the grid.

view of scene

Arrow keys allow you to move closer to or further away from the scene. The *up arrow* key moves your viewpoint towards the grid. The *down arrow* key moves your viewpoint away from the grid.

BUGS

It is possible to choose points not on the grid.

AUTHOR

Thant Tessman

HARDWARE REQUIREMENTS

Eight bitplanes and 2 Megabytes of memory are necessary.