sun microsystems

Software Technical Bulletin
August 1987

Software Information Services

Part Number 812-8701-07
Issue 1987 – 7
August 1987

sun® microsystems

Software Technical Bulletin
August 1987

Software Information Services

Part Number 812-8701-07
Issue 1987 – 7
August 1987

Software Technical Bulletins are distributed to customers with software/hardware or software only support contracts. Send comments or corrections to 'Software Technical Bulletins' at Sun Microsystems, Inc., 2550 Garcia Ave., M/S 2-312, Mountain View, CA 94043 or by electronic mail to *sun!stb-editor*. Customers who have technical questions about topics in the Bulletin should call Sun Customer Software Services AnswerLine at
800 USA-4-SUN.

Contents

1

NOTES & COMMENTS

1

NOTES & COMMENTS

Editor's Notes

Editor's Notes

The August editor's notes for the Software Technical Bulletin (STB) include an announcement of the current Sun software products and release levels table, notes on Customer Distributed BugsList (CDB) index entries, and a `shar` archive for calling NeWS from C.

Current Sun Software Products and Release Levels Table

The Software Technical Bulletin (STB) now includes a new feature. Each month you will see a table containing a list of current Sun software products. The current release level is shown for each product.

Use this table along with STB articles that appear in one or two issues after a new current release is available for a particular product. You can then better determine what your software needs are, what functions are available in a new release, and whether the release you are using is down-level from the most current product release.

You may note that the SunAlis release 2.0 article (July STB, page 249) and the SunINGRES release 5.0 article (July STB, page 254) describe the current release shown in the software products table.

The table will appear monthly in the 'NOTES & COMMENTS' section, following the "Editor's Notes."

Customer Distributed BugsList Cumulative Index

Note that this August STB issue contains the second Customer Distributed BugsList (CDB) that is included in the STB cumulative index.

For your convenience, the CDB contains similar index entries from one list to the next. In all, four such CDBs will be contained in a full year's cumulative index. Using similar index entries allows you to find up to four occurrences of the same type of bug per year.

You can find bugs in the index in two ways. First, look under 'bugs', an index entry that contains within it an alphabetized list of all CDB categories, topics, and Sun products. All entries under any topic within 'bugs' is found in one of the four yearly CDBs. Second, look in the index under the category, topic, or Sun product. You will then see an alphabetized list containing either 'bugs' or a Sun product name, followed by 'bugs'.

For example, you will find Datacomm listed under 'bugs', with a bug found on page 103. You will also find several specific bugs listed in the index under the entry 'Datacomm'.

The Hackers' Corner

You will find an article on using `screendump` for use with rasterfiles.

Again, please note that such applications, scripts, or code are not offered as released Sun products, but as items of interest to enthusiasts wanting to try out something for themselves. They may not not work in all cases, and may not be compatible with future SunOS releases. Please consult your local shell script or programming expert regarding any application, script, or code problems.

Sun Software Produce Releases

Current Software Sun Products and Release Levels

Product Name	Current Release
SunOS	3.4
Cross Compiler	1.0
SunLink BSC3270	4.0
SunLink Local 3270	4.0
SunLink SNA3270	4.0
SunLink IR	4.0
SunLink DDN	4.0
SunLink DNI	4.0
SunLink OSI	4.0
SunLink TE100	4.0
SunLink X.25	4.0
NeWS	1.0
Sun Common Lisp	2.0
Modula-2	1.0
SunAlis	2.0
SunGKS	2.0
SunINGRES	5.0
SunSimplify	1.0
SunUNIFY	2.0
Transcript	2.0

Current Sun Software Products and Release Levels

The table appearing below contains a list of current Sun software products and their respective current release levels.

You will note that the Software Technical Bulletin (STB) contains articles from time to time that detail technical changes in a given software product's next available release.

Please contact your sales representative if you decide that you would like to update the release level of a Sun software product you already use, or wish to purchase another product. Use the table below to determine whether your release is the current release level.

This table appears monthly in the STB for your convenience.

2

ARTICLES

2

ARTICLES

Other Mail Systems

Mail Systems on Sun Workstations: An Overview

Several approaches to processing electronic mail may be ported to Sun workstations in a reasonably straight-forward way. This article contains an overview of some mail 'systems' in wide usage at this time.

There are two very different kinds of software that process electronic mail. First, the *user agent* software offers an interface to the user to read, compose, and send mail. Second, the *mail transport system* software is a network service that transmits the mail from one location to another.

Sun software includes two user agents and one mail transport system. The first user agent is the Berkeley Mail program as modified by AT&T, which they call `mailx`. The second user agent is `mailtool`, which is a SunView program that uses the Berkeley Mail program to process mail.

The mail transport system is `sendmail`. It has been used so widely that it is quite solid by now. For example, machines expanding typical mailing lists often send 50,000 mail messages daily.

Other User Agents

Other user agents may be ported to Sun workstations. These include the MH-Rand system, the mail user's shell (MUSH), ELM, and various EMACS macros, most of which are in the public domain.

MUSH may be used in three ways: `suntool` mode, `tty`-based mode, and curser mode. MUSH has been tested and is working on machines including the VAX (4.2, and 4.3 BSD), PDP-11s (2.9 BSD), and the IBM-PC AT running System V (Xenix).

MUSH has a `csh` flavor of command line parsing in the `tty`-based mode and supports a subset from each of the following aspects of `csh` shown below.

- command history
- command line aliasing
- a 'piping' mechanism to redirect command input and output

MUSH fully emulates the Berkeley Mail program. Aliasing of commands is useful. There is a `.mushrc` file in your home directory, very much like the Berkeley Mail `.mailrc` file. MUSH has multiple levels of built-in help so that each command can be understood completely, without having to refer to the manual.

Other Transport Systems

Other transport systems include MMDF used by CSNET, the UUCP Project's `smail` program, as well as `uumail` and other UUCP-oriented programs.

For Further Information

See this month's STB In Depth feature on the family of TCP/IP protocols. This article, written by Charles L. Hedrick from Rutgers University, is reprinted with permission. It provides an introduction to protocols widely used to transport mail, as well as references for further information on specific topics.

Saving Disk Space

Saving Disk Space

This article contains some suggestions on how to save disk space, and certain system performance degradations that may occur in some cases. You can then decide whether the tradeoff is useful in your application.

The Initial Set Up

Consider one installation of a Sun 3/260 with 10 Sun3/50 diskless nodes, all using two 280 Mbyte disks. To save disk space, a `root` partition was set up on the server with approximately 25 Mbytes of disk space. A `root` file partition was then set up for each of the clients. On the server, subdirectories of `/tmp` were made for each client using the client hostnames; `client1`, `client2`, `client3`, and so forth.

The equivalent of the line shown below was added to the `/etc/fstab` for each of the clients. The example shown is for client1.

```
server:/tmp/client1 /tmp nfs rw,hard 0 0
```

Note that `/tmp` on the client remains as a directory so that if the `mount` failed, you would still be able to use it. This also applies in the case of booting single user.

This set up results in each of the 10 diskless nodes and the server having access to a `/tmp` which is about 15 - 20 Mbytes. You would otherwise use more than another 100 Mbytes.

Problems and Solutions

One problem is that the client does not have `root` access to the server file system, unless you have patched the kernel on the server to translate `root` across the network. To see of this upsets the application, use the experiment shown below.

```
% su
# touch /tmp/filename
# chown fred /tmp/filename
```

A second problem to look for is a possible significant negative performance impact on the diskless clients when the `/tmp` partition is mounted via NFS. Client activity that uses `/tmp` frequently, especially activity involving frequent file creation and deletion, will become slower. Whether or not this is a noticeable problem depends heavily on tasks the clients are processing, and varies from one application to another.

One more improvement is to make a partition mounted onto `/tmp` of the server and clients. For the server and each client, make a subdirectory in the mounted partition. Then symbolically link `/tmp` for each to the directory. This results in making the server's `root` partition small and sharing that space for all. Also, it might be on a separate disk and give even better performance.

Summary

A summary appears below for saving disk space for many clients on a server, by NFS mounting a subdirectory of the server `/tmp` as the client `/tmp`.

- Some programs need root privileges to write to the server's file system. A patch should be applied to the server kernel. The patch is described in Chapter 2, 'Sun Network Services', of the manual *System Administration for the Sun Workstation*, part number 800-1323.
- Scripts that automatically cleanup the `/tmp` directory when the machine reboots or via `cron`, should not do a recursive directory cleanup, but just the top level.
- Better performance may be obtained by not putting all of the `/tmp` directories on the `root` partition, but by directing to `/usr/tmp/client[0-9]`, and by giving the `root` partition on the server a small size as well. The `/tmp` redirection could also be to areas on a second disk.
- Performance may be impacted by mounting `/tmp` for clients via NFS instead of by ND.

screendumping Rasterfiles

screendump: Saving and Printing Images

Customer Support receives many questions on how to save screen images to files for subsequent display or hardcopy output. This article summarizes usage and possible pitfalls of the related utilities on the Sun workstation.

Saving Images

Type the following command to save a screen image to a file named `rasterfile`.

```
% screendump rasterfile
```

If you now run *file(1)* on `rasterfile` you will see a response something like the example shown below.

```
% file rasterfile
rasterfile: rasterfile, 1152x900x1 standard format image
```

file(1) is reporting the width, height and depth of the raster image in pixels, respectively. If this image had been created on a color screen, the depth would have been listed as '8'. For those interested in the details of rasterfile format, see the *Pixrect Reference Manual*, part number 800-1254, and `/usr/include/rasterfile.h`.[2]

Printing Saved Images

Once an image has been saved, it is available for redisplay or printing. To display the image on the screen, simply use *screenload(1)* as shown below.[3]

```
% screenload rasterfile
```

To print `rasterfile` on the Sun Laserwriter, a special TRANSCRIPT[4] filter has been created for use by `lpr`. Thus, the command shown below will access this filter and print `rasterfile`.

```
% lpr -v rasterfile
```

Printing can be a problem with color images, however. Since the Laserwriter represents only monochrome images, you must first convert a color rasterfile before sending it to the printer. A handy filter for this purpose is *rasfilter8to1(1)*.

[2] In particular, SunView and Pixrect programmers will want to know that `screendump` creates rasterfiles with a colormap size of 256. SunCore programmers should be aware that the `screendump` colormap takes color intensity values from zero to 255 rather than from zero to one as in SunCore. SunCore programs need to convert from one colormap type to the other to interface with `screendump`.

[3] The SunView, Pixrect, and SunCore packages all have internal facilities for loading a rasterfile as well. See *pr_load()* in the Pixrect Reference Manual, and *file_to_raster()* in the SunCore Reference Manual, part number 800-1257, for details.

[4] TRANSCRIPT is a trademark of Adobe Systems, Inc.

Use the command shown below to print an image created on a color monitor.

```
% rasfilter8to1 < rasterfile | lpr -v
```

In addition, `screendump` on color monitors can fail, causing a corrupted image, if anything on the screen is moving during the dump. Examples include moving the mouse cursor, a ticking clock, a perfmeter (performance meter), or even a blinking caret. So be sure to exit or cover any tool which is updating the screen while `screendump` is active.

Another special case is the Sun high-resolution, monochrome monitor. If a screendump made on this monitor is printed, part of the image will be cut off. The solution is to access the TRANSCRIPT filter explicitly, telling it to scale the image so as to fit all on one page. Try the command shown below to scale the image to fit on an eight-inch page.

```
% screendump | pssun -S 8 | lpr
```

Note that since the filter `pssun` is being called explicitly, the `-v` option to `lpr` should be omitted. Calling the filter explicitly provides several additional capabilities in the form of options to `pssun`, including rotation and replication. See the *pssun(1)*[5] manual page for more information.

Current Frame Buffer Details

Details follow describing how `screendump` knows to dump the current frame buffer. `screendump` has a default notion of the frame buffer, `/dev/fb`. However, there are cases in which one wants to dump images from a frame buffer other than `/dev/fb`, for example, on a machine with two frame buffers. For this purpose, `screendump` has a `-f` option which, in the example shown below, will store the image currently displayed on the color frame buffer, even if that frame buffer is not represented by `/dev/fb`. Note that `screenload` has a parallel `-f` option.

```
% screendump -f /dev/cgtwo0 rasterfile
```

Another reason to specify the frame buffer name is the unique frame buffer configuration of the Sun-3/110. This machine has two frame buffers, one monochrome (`/dev/bwtwo0`) and one color (`/dev/cgfour0`), plus an overlay plane. The *switcher(1)* man page describes how to invoke `suntools` twice, once on each frame buffer. When starting `suntools` in this way, be sure to use the `-f` option of `screenload` or `screendump` to access the desired frame buffer.

Some users like to start `suntools` on the 'merged' frame buffer, that is, combining the color and monochrome frame buffers and using the overlay plane to indicate which frame buffer is being used on a per-pixel basis. Note that starting `suntools` without any options means running in the merged frame

[5] This manual page must be loaded from the TRANSCRIPT installation tape.

buffer. Color windows, e.g. color images from a graphics package, reside in `cgfour0` and monochrome windows, e.g. `shelltools`, `cmdtools`, `textedit` windows, and the like, reside in `bwtwo0`.

As a result, it is not currently possible to screendump the entire screen in this particular case. Running `screendump` on `cgfour0` saves the color images while specifying `bwtwo0` stores the monochrome. In either case, the parts of the screen defined by the other frame buffer are left blank in the resulting rasterfile. If no frame buffer is specified, `/dev/fb` is used and this corresponds to `/dev/cgfour0`.

For Further Information

UNIX manual pages used as references in preparing this article are listed below.

- *screendump(1)*
- *screenload(1)*
- *rasfilter8to1(1)*
- *pssun(1)*
- *rastrepl(1)*
- *file(1)*
- *lpr(1)*
- *switcher(1)*
- *suntools(1)*

Sun manuals used as references are shown below.

- *Pixrect Reference Manual*, part number 800-1254
- *SunView Programmer's Guide*, part number 800-1345
- *SunCore Reference Manual*, part number 800-1257
- *Release 3.2 Manual for the Sun Workstation*, part number 800-1364
- *Release 3.4 Manual for the Sun Workstation*, part number 800-1614

The following `include` file is used for reference.

- `/usr/include/rasterfile.h`

3

STB SHORT SUBJECTS

3

STB SHORT SUBJECTS

Using *stb!hotline*

Using your ***sun!hotline***

The email address *sun!hotline* is for use by customers holding software support contracts. When sending email to *sun!hotline*, you should always include the information listed below. If any of the information is missing, it may take longer to get the needed support.

- workstation model and serial number
- name
- phone number and area code
- electronic mail address (`sendmail` does not always show the correct address.)
- company or organization name and address
- SunOS release number (See the June STB short subject, page 205, to find how to determine your SunOS release level.)
- any information which may help diagnose the problem

Information that helps diagnose the problem includes what was running when the problem occurred, a description of symptoms including exact text of any error messages, a small test program that exhibits the problem, your hardware configuration, or any other information that seems appropriate to the problem at hand.

Please avoid sending large files by email.

A service call is logged and the call is assigned to an engineer. After the call is logged, you will receive email giving the service order (SO) number and an engineer's phone number to call. This usually is done the same day the email is received.

Use the phone number when you have not received a response from an engineer within 24 hours. Please note that a response is not necessarily an answer. The engineer may simply send email just to say 'I have it and am working on it.'

Finally, please note that *sun!hotline* is for software-related questions only. For customers holding support contracts, hardware troubleshooting questions may be called in to **800 USA-4-SUN**.

Use Your USAC

U.S. Answer Center: Feedback Wanted

Here in the U.S. Answer Center (USAC), we strive to maintain the highest level of commitment to our customers, listening and responding to their needs. In order to help us improve our service, we encourage your feedback. Please send all letters to the address shown below.

```
Sun Microsystems, Inc.
2550 Garcia Ave.
Mt. View, CA  94043
Attn: Marion Brown 2-30
```

Page Boundary Errors

Page Faults and Bus Errors

This short subject contains an overview of page faults, their causes, and errors reported.

Generically, a page fault occurs when the central processing unit (CPU) tries to access a page (i.e., an 8 kbyte portion of virtual memory) which is not currently valid. The memory management unit (MMU) detects the situation, and issues a bus error.

Most page faults occur because the kernel has paged out the relevant page, or has removed its page map entry. Some page faults take place because there is no such page in the virtual address space for the process.

The kernel keeps the complete set of page table entries for each process in memory. If it finds that the fault took place while the physical page is still available, it maps it back in. If the page has been swapped out to disk, it brings it back in and then maps it in. If the page does not exist, it issues SIGSEGV, and the process dies of a segmentation violation.

In most cases, the kernel panics when it tries to access a page in its own address space that is not there. The kernel takes a page fault if the page is not there in the case when the kernel touches pages in the user's address space, e.g. when it is doing a `copyin`, `copyout`, `uiomove`, and the like.

Kernel Page Faults without Errors

The kernel takes a page fault if the page is not there in the case when the kernel touches pages in the user's address space, e.g. when it is doing a `copyin`, `copyout`, `uiomove`, and the like.

After the page fault, the kernel starts a `pagein` and blocks the process until the page comes in, and then resumes the kernel code at the point of the `pagein`. Note that this means that no kernel code that copies data to or from the user's address space can be considered atomic and non-preemptible.

Also note that some operating systems (e.g., DEC's VAX-VMS) page some of their own memory. A kernel-mode page fault is therefore not a bad event in all operating systems.

4

IN DEPTH

4

IN DEPTH

Internet Protocols

Introduction to the Internet Protocols

This is an introduction to the Internet networking protocols (TCP/IP). It includes a summary of the facilities available and brief descriptions of the major protocols in the family.[6]

This document is an introduction to the transmission control protocol (TCP) and the Internet protocol (IP), followed by advice on what to read for more information. This is not intended to be a complete description. It can give you a reasonable idea of the capabilities of the protocols. Throughout the text, you will find references to the standards, in the form of request for comment (RFC) or IEN numbers. These are document numbers. The final section of this document tells you how to get copies of those standards.

1: What is TCP/IP?

TCP/IP is a set of protocols developed to allow cooperating computers to share resources across a network. It was developed by a community of researchers centered around the ARPAnet. Certainly the ARPAnet is the best-known TCP/IP network. However as of June 1987, at least 130 different vendors had products that support TCP/IP, and thousands of networks of all kinds use it.

First some basic definitions. The most accurate name for the set of protocols we are describing is the *Internet protocol suite*. TCP and IP are two of the protocols in this suite. Because TCP and IP are the best known of the protocols, it has become common to use the term TCP/IP or IP/TCP to refer to the whole family. However this can lead to some oddities. For example, one can talk about NFS as being based on TCP/IP, even though it does not use TCP at all. It does use IP. But it uses an alternative protocol, UDP, instead of TCP.

[6] Copyright (C) 1987, Charles L. Hedrick. Anyone may reproduce this document, in whole or in part, provided that: (1) any copy or republication of the entire document must show Rutgers University as the source, and must include this notice; and (2) any other use of this material must reference this manual and Rutgers University, and the fact that the material is copyrighted by Charles Hedrick and is used by permission.

The Internet is a collection of networks, including the ARPAnet, NSFnet, regional networks such as NYsernet, local networks at a number of university and research institutions, and a number of military networks. The term *Internet* applies to this entire set of networks. The subset of them that is managed by the Department of Defense is referred to as the Defense Data Network (DDN). This includes some research-oriented networks, such as the ARPAnet, as well as more strictly military ones. Because much of the funding for Internet protocol developments is done via the DDN organization, the terms Internet and DDN can sometimes seem equivalent.

All of these networks are connected to each other. Users can send messages from any of them to any other, except where there are security or other policy restrictions on access. Officially speaking, the Internet protocol documents are simply standards adopted by the Internet community for its own use. More recently, the Department of Defense issued a MILSPEC definition of TCP/IP. This was intended to be a more formal definition, appropriate for use in purchasing specifications. However, most of the TCP/IP community continues to use the Internet standards. The MILSPEC version is intended to be consistent with it.

Thus, TCP/IP is a family of protocols. A few provide 'low-level' functions needed for many applications. These include IP, TCP, and the user datagram protocol (UDP). Others are protocols for doing specific tasks, e.g. transferring files between computers, sending mail, or finding out who is logged in on another computer. Initially TCP/IP was used mostly between minicomputers or mainframes. These machines had their own disks, and generally were self-contained. Thus the most important 'traditional' TCP/IP services are described below.

File Transfer

The file transfer protocol (FTP) allows a user on any computer to get files from another computer, or to send files to another computer. Security concerns require the user to specify a user name and password for the other computer. Provisions are made for processing file transfers between machines with different character sets, end-of-line conventions, and the like. This is not quite the same thing as more recent *network file system* or *netBIOS* protocols, which will be described below. Rather, FTP is a utility that you run any time you want to access a file on another system. You use it to copy the file to your own system. You then work with the local copy. See RFC 959 for specifications for FTP.

Remote Login

The network terminal protocol (TELNET) allows a user to log in on any other computer on the network. You start a remote session by specifying a computer to connect to. From that time until you finish the session, anything you type is sent to the other computer. Note that you are really still talking to your own computer. But the `telnet` program effectively makes your computer invisible while it is running.

Every character you type is sent directly to the other system. Generally, the connection to the remote computer behaves much like a dialup connection. That is, the remote system will ask you to log in and give a password, in whatever manner it would normally ask a user who had just dialed it up. When you log off of the other computer, the `telnet` program exits, and you will find yourself talking to your own computer. Microcomputer implementations of `telnet` generally include a terminal emulator for some common types of terminals. See RFCs 854 and 855 for specifications for `telnet`. Note that the `telnet` protocol should not be confused with Telenet, a vendor of commercial network services.

Computer Mail

This allows you to send messages to users on other computers. Originally, people tended to use only one or two specific computers. They would maintain mail files on those machines. The computer mail system is simply a way for you to add a message to another user's mail file. There are some problems with this in an environment where microcomputers are used. The most serious is that a microcomputer is not well suited to receive computer mail.

When you send mail, the `mail` software expects to be able to open a connection to the addressee's computer, in order to send the mail. If this is a microcomputer, it may be turned off, or it may be running an application other than the mail system. For this reason, mail is normally processed by a larger system, where it is practical to have a mail server running all the time. Microcomputer mail software then becomes a user interface that retrieves mail from the mail server.

See RFCs 821 and 822 for specifications for computer mail. See RFC 937 for a protocol designed for microcomputers to use in reading mail from a mail server.

These services should be present in any implementation of TCP/IP, except that micro-oriented implementations may not support computer mail. These traditional applications still play a very important role in TCP/IP-based networks. However, more recently, the way in which networks are used has been changing. The older model of a number of large, self-sufficient computers is beginning to change. Now many installations have several kinds of computers, including microcomputers, workstations, minicomputers, and mainframes. These computers are likely to be configured to perform specialized tasks. Although people are still likely to work with one specific computer, that computer will call on other systems on the net for specialized services.

This has led to the *server/client* model of network services. A server is a system that provides a specific service for the rest of the network. A client is another system that uses that service. Note that the server and client need not be on different computers. They could be different programs running on the same computer.

The kinds of servers typically present in a modern computer setup are described below. Note that these computer services can all be provided within the framework of TCP/IP.

Network File Systems

This allows a system to access files on another computer in a somewhat more closely integrated fashion than FTP. A network file system provides the illusion that disks or other devices from one system are directly connected to other systems. There is no need to use a special network utility to access a file on another system. Your computer simply thinks it has some extra disk drives. These extra 'virtual' drives refer to the other systems' disks. This capability is useful for several different purposes. It lets you put large disks on a few computers, but still give others access to the disk space.

Aside from the obvious economic benefits, this allows people working on several computers to share common files. It makes system maintenance and backup easier, because you do not have to worry about updating and backing-up copies many different machines. A number of vendors now offer high-performance, diskless computers. These computers have no disk drives at all. They are entirely dependent upon disks attached to common *file servers.*

See RFCs 1001 and 1002 for a description of PC-oriented NetBIOS over TCP. In the workstation and minicomputer area, Sun Microsystem's Network File System (NFS) is more likely to be used. Protocol specifications for it are available from Sun Microsystems, Inc.

Remote Printing

This allows you to access printers on other computers as if they were directly attached to yours. The most commonly used protocol is the remote lineprinter protocol from Berkeley UNIX. Unfortunately, there is no protocol document for this. However, the C code is easily obtained from Berkeley, so implementations are common.

Remote Execution

This allows you to request that a particular program be run on a different computer. This is useful when you can do most of your work on a small computer, but a few tasks require the resources of a larger system. There are a number of different kinds of remote execution. Some operate on a command-by-command basis. That is, you request that a specific command or set of commands should run on some specific computer. More sophisticated versions will choose a system that happens to be free. However, there are also *remote procedure call* systems that allow a program to call a subroutine that will run on another computer.

There are many protocols of this sort. Berkeley UNIX contains two servers to execute commands remotely: `rsh` and `rexec`. The man pages describe the protocols that they use. The user-contributed software with Berkeley 4.3 contains a *distributed shell* that distributes tasks among a set of systems, depending upon load. Remote procedure call mechanisms have been a topic for research for a number of years, so many organizations have implementations of such facilities. The most widespread, commercially-supported remote procedure call protocols (RPCs) seem to be Xerox's Courier and Sun Microsystem's RPC. Protocol documents are available from Xerox and Sun Microsystems. There is a public implementation of Courier over TCP as part of the user-contributed software with Berkeley 4.3. An implementation of RPC was posted to Usenet by Sun Microsystems, and also appears as part of the user-contributed software with Berkeley 4.3.

Name Servers

In large installations, there are a number of different collections of names that have to be managed. This includes users and their passwords, names and network addresses for computers, and accounts. It becomes tedious to keep this data up-to-date on all of the computers. Thus the databases are kept on a small number of systems. Other systems access the data over the network.

RFCs 822 and 823 describe the name server protocol used to keep track of host names and Internet addresses on the Internet. This is now a required part of any TCP/IP implementation. IEN 116 describes an older name server protocol that is used by a few terminal servers and other products to look up host names. Sun Microsystem's Yellow Pages system is designed as a general mechanism to process user names, file sharing groups, and other databases commonly used by UNIX systems. It is widely available commercially. Its protocol definition is available from Sun Microsystems.

Terminal Servers

Many installations no longer connect terminals directly to computers. Instead they connect them to terminal servers. A terminal server is simply a small computer that only knows how to run `telnet` or some other protocol to do remote login. If your terminal is connected to one of these, you simply type the name of a computer, and you are connected to it. Generally it is possible to have active connections to more than one computer at the same time. The terminal server will have provisions to switch between connections rapidly, and to notify you when output is waiting for another connection. Terminal servers use the `telnet` protocol, already mentioned. However, any real terminal server will also have to support name service and a number of other protocols.

Network-Oriented Window Systems

Until recently, high-performance graphics programs had to execute on a computer that had a bit-mapped graphics screen directly attached to it. Network window systems allow a program to use a display on a different computer. Full-scale network window systems provide an interface that lets you distribute tasks to the systems that are best suited to process them, but still give you a single graphically-based user interface. The most widely-implemented window system is X. A protocol description is available from MIT's Project Athena. A reference implementation is publically available from MIT. A number of vendors are also supporting NeWS, a window system defined by Sun Microsystems. Both of these systems are designed to use TCP/IP.

Note that some of the protocols described above were designed by Berkeley, Sun Microsystems, or other organizations. Thus they are not officially part of the Internet protocol suite. However, they are implemented using TCP/IP, just as normal TCP/IP application protocols are. Since the protocol definitions are not considered proprietary, and since commercially-support implementations are widely available, it is reasonable to think of these protocols as being effectively part of the Internet suite. Note that the list above is simply a sample of the sort of services available through TCP/IP. However, it does contain the majority of the 'major' applications. The other commonly-used protocols tend to be specialized facilities for getting information of various kinds, such as who is logged in, the time of day, and so forth. However, if you need a facility that is not listed here, look through the current edition of 'Internet Protocols', currently RFC 1011. It lists all of the available protocols, and also to look at some of the major TCP/IP implementations to see what various vendors have added.

2: General Description of the TCP/IP Protocols

TCP/IP is a layered set of protocols. In order to understand what this means, it is useful to look at an example. A typical situation is sending mail. First, there is a protocol for `mail`. This defines a set of commands which one machine sends to another, e.g. commands to specify who the sender of the message is, who it is being sent to, and then the text of the message. However, this protocol assumes that there is a way to communicate reliably between the two computers. `mail`, like other application protocols, simply defines a set of commands and messages to be sent. It is designed to be used together with TCP and IP.

TCP is responsible for making sure that the commands get through to the other end. It keeps track of what is sent, and retransmits anything that did not get through. If any message is too large for one datagram, e.g. the text of the mail, TCP will split it up into several datagrams, and make sure that they all arrive correctly. Since these functions are needed for many applications, they are put together into a separate protocol, rather than being part of the specifications for sending mail. You can think of TCP as forming a library of routines that applications can use when they need reliable network communications with another computer. Similarly, TCP calls on the services of IP.

Although the services that TCP supplies are needed by many applications, there are still some kinds of applications that do not need them. However, there are some services that every application needs. So these services are put together into IP. As with TCP, you can think of IP as a library of routines that TCP calls on, but which is also available to applications that do not use TCP. This strategy of building several levels of protocol is called *layering*. We think of the applications programs such as mail, TCP, and IP, as being separate *layers*, each of which calls on the services of the layer below it. Generally, TCP/IP applications use the four layers described below.

- an application protocol such as `mail`
- a protocol such as TCP that provides services needed by many applications
- IP, which provides the basic service of getting datagrams to their destinations
- the protocols needed to manage a specific physical medium, such as Ethernet or a point-to-point line

TCP/IP is based on the *catenet model.* This is described in more detail in IEN 48. This model assumes that there are a large number of independent networks connected together by gateways. The user should be able to access computers or other resources on any of these networks. Datagrams will often pass through a dozen different networks before getting to their final destinations. The routing needed to accomplish this should be completely invisible to the user.

As far as the user is concerned, all she or he needs to know in order to access another system is an Internet address. This is an address that looks like 128.6.4.194. It is actually a 32-bit number. However, it is normally written as four decimal numbers, each representing eight bits of the address.

The term *octet* is used by Internet documentation for such 8-bit sections. The term 'byte' is not used, because TCP/IP is supported by some computers that have byte sizes other than eight bits. Generally, the structure of the address gives you some information about how to get to the system. For example, 128.6 is a network number assigned by a central authority to Rutgers University. Rutgers uses the next octet to indicate which of the campus Ethernets is involved. 128.6.4 is an Ethernet used by the Computer Science Department. The last octet allows for up to 254 systems on each Ethernet. It is 254 because 0 and 255 are not allowed, for reasons that will be discussed later. Note that 128.6.4.194 and 128.6.5.194 would be different systems. The structure of an Internet address is described in more detail later.

People normally refer to systems by name, rather than by their Internet addresses. When we specify a name, the network software looks it up in a database, and finds the corresponding Internet address. Most of the network software deals strictly in terms of the address. RFC 882 describes the name server technology used to process this lookup.

TCP/IP is built on *connectionless* technology. Information is transferred as a sequence of *datagrams*. A datagram is a collection of data that is sent as a single message. Each of these datagrams is sent through the network individually. There are provisions to open connections, i.e. to start a conversation that will continue for some time. However, at some level, information from those connections is broken-up into datagrams, and those datagrams are treated by the network as completely separate.

For example, suppose you want to transfer a 15,000-octet file. Most networks can not process a 15,000-octet datagram. So the protocols will break this up into something like 30 separate, 500-octet datagrams. Each of these datagrams will be sent to the other end. At that point, they will be put back together into the 15,000-octet file. However, while those datagrams are in transit, the network does not know that there is any connection between them. It is possible that datagram 14 will actually arrive before datagram 13. It is also possible that somewhere in the network, an error will occur, and some datagram will not get through at all. In that case, that datagram has to be sent again.

Note that the terms *datagram* and *packet* often seem to be nearly interchangeable. Technically, datagram is correct to use when describing TCP/IP. A datagram is a unit of data, which is what the protocols process. A packet is a physical object, appearing on an Ethernet or some wire. In most cases a packet simply contains a datagram, so there is very little difference. However, they can differ. When TCP/IP is used on top of X.25, the X.25 interface breaks-up the datagrams into 128-byte packets. This is transparent to IP, because the packets are put back together into a single datagram at the other end before being processed by TCP/IP. So in this case, one IP datagram would be carried by several packets. However, with most media, there are efficiency advantages to sending one datagram per packet, and so the distinction tends to vanish.

2.1: The TCP Level

Two separate protocols are involved in processing TCP/IP datagrams. TCP is responsible for breaking-up the message into datagrams, reassembling them at the other end, resending anything that gets lost, and putting things back in the right order. IP is responsible for routing individual datagrams. It may seem like TCP is doing all the work. And in small networks that is true. However, in the Internet, simply getting a datagram to its destination can be a complex task.

For example, connection may require the datagram to go through several networks at Rutgers, a serial line to the John von Neuman Supercomputer Center, a couple of Ethernets there, a series of 56 Kbaud phone lines to another NSFnet site, and more Ethernets on another campus. Keeping track of the routes to all of the destinations and processing incompatibilities among different transport media turns out to be a complex task. Note that the interface between TCP and IP is fairly simple. TCP simply hands IP a datagram with a destination. IP does not know how this datagram relates to any datagram before it or after it.

Clearly it is not enough to get a datagram to the right destination. TCP has to know which connection this datagram is part of. This task is referred to as *demultiplexing*. In fact, there are several levels of demultiplexing found in TCP/IP.

The information needed to do this demultiplexing is contained in a series of headers. A header is a few extra octets added to the beginning of a datagram by some protocol in order to keep track of it. It is a lot like putting a letter into an envelope and putting an address on the outside of the envelope. Except with modern networks it happens several times. It is like you put the letter into a little envelope, your administrator puts that into a somewhat bigger envelope, the campus mail center puts that envelope into a still bigger one, and so forth.

Header Overview

An overview of the headers that get added to a message that passes through a typical TCP/IP network follows. We start with a single data stream, say a file you are trying to send to some other computer as shown below.

TCP breaks it up into manageable units. In order to do this, TCP has to know how large a datagram your network can process. Actually, the TCPs at each end say how large a datagram they can process, and then they pick the smallest size.

TCP puts a header at the front of each datagram. This header contains at least 20 octets, but the most important ones are a source and destination *port number* and a *sequence number*. The port numbers are used to keep track of different conversations. Suppose three different people are transferring files. Your TCP might allocate port numbers 1000, 1001, and 1002 to these transfers. When you are sending a datagram, this becomes the 'source' port number, since you are the source of the datagram. Of course, the TCP at the other end has assigned a port number of its own for the conversation.

Your TCP has to know the port number used by the other end as well. It finds out when the connection starts, as we will explain below. It puts this in the 'destination' port field. Of course, if the other end sends a datagram back to you, the source and destination port numbers will be reversed, since then it will be the source and you will be the destination.

Each datagram has a sequence number. This is used so that the other end can make sure that it gets the datagrams in the right order, and that it has not missed any. See the TCP specification for details. TCP does not number the datagrams, but the octets. So if there are 500 octets of data in each datagram, the first datagram might be numbered 0, the second 500, the next 1000, the next 1500, and so forth.

Checksum is a number that is computed by adding up all the octets in the datagram. See the TCP specification for details. The result is put in the header. TCP at the other end computes the checksum again. If they disagree, then something bad happened to the datagram in transmission, and it is discarded. The datagram now appears as shown below.

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Source Port          |       Destination Port        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Sequence Number                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Acknowledgment Number                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Data |           |U|A|P|R|S|F|                               |
| Offset| Reserved  |R|C|S|S|Y|I|            Window             |
|       |           |G|K|H|T|N|N|                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Checksum            |         Urgent Pointer        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   your data ... next 500 octets                               |
|   ......                                                      |
```

If we abbreviate the TCP header as 'T', then the whole file now looks as shown below.

```
T....   T....   T....   T....   T....   T....   T....
```

Note that there are items in the header not described above. They are generally involved with managing the connection. In order to make sure the datagram has arrived at its destination, the recipient has to send back an *acknowledgement*. This is a datagram whose 'acknowledgement number' field is filled in.

For example, sending a packet with an acknowledgement of 1500 indicates that you have received all the data up to octet number 1500. If the sender does not get an acknowledgement within a reasonable amount of time, it sends the data again. The window is used to control how much data can be in transit at any one time. It is not practical to wait for each datagram to be acknowledged before sending the next one. That would slow processing too much. On the other hand, you can not just keep sending, or a fast computer might overrun the capacity of a slow one to absorb data. Thus each end indicates how much new data it is currently prepared to absorb by putting the number of octets in its *window* field.

As the computer receives data, the amount of space left in its window decreases. When it goes to zero, the sender has to stop. As the receiver processes the data, it increases its window, indicating that it is ready to accept more data. Often the same datagram can be used to acknowledge receipt of a set of data and to give permission for additional new data, by an updated window. The *urgent* field allows one end to tell the other to skip ahead in its processing to a particular octet. This is often useful for handling asynchronous events, for example when you type a control character or other command that interrupts output. The other fields are beyond the scope of this document.

2.2: The IP Level

TCP sends each of these datagrams to IP. Of course, it has to tell IP the Internet address of the computer at the other end. Note that this is the only IP concern. It does not care about what is in the datagram, or even in the TCP header. The IP task is to find a route for the datagram and get it to the other end. In order to allow gateways or other intermediate systems to forward the datagram, it adds its own header.

The main items in this header are the source and destination Internet address (32-bit addresses, like 128.6.4.194), the protocol number, and another checksum. The source Internet address is the address of your machine. This is necessary so the other end knows where the datagram came from. The destination Internet address is the address of the other machine. This is necessary so any gateways in the middle know where you want the datagram to go.

The protocol number tells the IP at the other end to send the datagram to TCP. Although most IP traffic uses TCP, there are other protocols that can use IP, so you have to tell IP which protocol to send the datagram to. Finally, the checksum allows IP at the other end to verify that the header was not damaged in transit. Note that TCP and IP have separate checksums. IP needs to be able to verify that the header did not get damaged in transit, or it could send a message to the wrong place. For reasons beyond the scope of this document, it is both more efficient and safer to have TCP compute a separate checksum for the TCP header and data. Once IP has added its header, the message appears as shown below.

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version|  IHL  |Type of Service|          Total Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Identification        |Flags|      Fragment Offset    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Time to Live |    Protocol   |         Header Checksum       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Source Address                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Destination Address                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  TCP header, then your data ......                            |
|                                                               |
```

If we represent the IP header by an 'I', your file now appears as shown below.

```
IT....   IT....   IT....   IT....   IT....   IT....   IT....
```

Again, the header contains some additional fields that have not been discussed.

Most of them are beyond the scope of this document. The flags and fragment offset are used to keep track of the pieces when a datagram has to be split up. This can happen when datagrams are forwarded through a network for which they are too large. The *time-to-live* is a number that is decremented when the datagram passes through a system. When it goes to zero, the datagram is discarded. This is done in case a loop develops in the system. Of course, this should be impossible, but well-designed networks are built to cope with 'impossible' conditions.

At this point, it is possible that no more headers are needed. If your computer happens to have a direct phone line connecting it to the destination computer, or to a gateway, it may simply send the datagrams out on the line. However, it is more likely that a synchronous protocol such as HDLC would be used, and it would add at least a few octets at the beginning and end.

2.3: The Ethernet Level

Most networks use Ethernet. Ethernet has its own headers and addresses. The Ethernet designers wanted to make sure that no two machines would have the same Ethernet address. Furthermore, they did not want the user to be concerned with assigning addresses. So each Ethernet controller comes with an address built-in from the factory.

In order to make sure that they would never have to reuse addresses, the Ethernet designers allocated 48 bits for the Ethernet address. Ethernet equipment manufacturers have to register with a central authority, to make sure that the numbers they assign do not overlap any other manufacturer. Ethernet is a 'broadcast medium'. That is, it is in effect shared usage, like an old 'party line' telephone. When you send a packet out on the Ethernet, every machine on the network sees the packet. So something is needed to make sure that the right machine gets it.

This involves the Ethernet header. Every Ethernet packet has a 14-octet header that includes the source and destination Ethernet address, and a type code. Each machine is supposed to pay attention only to packets with its own Ethernet address in the destination field. It is possible to cheat, which is one reason that Ethernet communications are not secure.

Note that there is no connection between the Ethernet address and the Internet address. Each machine has to have a table of which Ethernet address corresponds to which Internet address. In addition to the addresses, the header contains a *type code*. The type code is to allow for several different protocol families to be used on the same network. So you can use TCP/IP, DECnet, Xerox NS, and so forth, at the same time. Each of them will put a different value in the type field.

Finally, there is a *checksum*. The Ethernet controller computes a checksum of the entire packet. When the other end receives the packet, it recomputes the checksum, and throws the packet away if the answer disagrees with the original. The checksum is put on the end of the packet, not in the header. The final result is such that your message appears as shown below.

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Ethernet destination address (first 32 bits)            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Ethernet dest (last 16 bits)  |Ethernet source (first 16 bits)|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Ethernet source address (last 32 bits)                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        Type code              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  IP header, then TCP header, then your data                   |
|                                                               |
    ...
|                                                               |
|   end of your data                                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Ethernet Checksum                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

If we represent the Ethernet header with 'E', and the Ethernet checksum with 'C', your file now is as shown below.

```
EIT....C   EIT....C   EIT....C   EIT....C   EIT....C
```

When these packets are received by the other end, the headers are removed. The Ethernet interface removes the Ethernet header and the checksum. It looks at the type code. Since the type code is the one assigned to IP, the Ethernet device driver passes the datagram up to IP. IP removes the IP header. It looks at the IP protocol field. Since the protocol type is TCP, it passes the datagram up to TCP. TCP now looks at the sequence number. It uses the sequence numbers and other information to combine all the datagrams into the original file.

For detailed descriptions of the items discussed here, see RFC 793 for TCP, RFC 791 for IP, and RFCs 894 and 826 for sending IP over Ethernet.

3: Well-Known Sockets and the Applications Layer

There needs to be a way for you to open a connection to a specified computer, log into it, tell it what file you want, and control the transmission of the file. If you have a different application in mind, e.g. computer mail, some analogous protocol is needed. This is done by *application protocols*. The application protocols run 'on top of' TCP/IP. That is, when they want to send a message, they give the message to TCP. TCP makes sure it gets delivered to the other end. Because TCP and IP take care of all the networking details, the applications protocols can treat a network connection as if it were a simple byte stream, like a terminal or phone line.

Finding an application is a complex process. Suppose you want to send a file to a computer whose Internet address is 128.6.4.7. To start the process, you need more than just the Internet address. You have to connect to the FTP server at the other end. In general, network programs are specialized for a specific set of tasks. Most systems have separate programs to process file transfers, remote terminal logins, mail, and the like.

When you connect to 128.6.4.7, you have to specify that you want to talk to the FTP server. This is done by having *well-known sockets* for each server. Recall that TCP uses port numbers to keep track of individual conversations. User programs normally use random port numbers. However, specific port numbers are assigned to the programs that sit waiting for requests.

For example, if you want to send a file, you will start a program called `ftp`. It will open a connection using some random number, for example, 1234, for the port number on its end. However it will specify port number 21 for the other end. This is the official port number for the FTP server. Note that there are two different programs involved. You run `ftp` on your side. This is a program designed to accept commands from your terminal and pass them on to the other end. The program that you talk to on the other machine is the FTP server. It is designed to accept commands from the network connection, rather than from an interactive terminal. There is no need for your program to use a well-known socket number for itself. Nobody is trying to find it. However, the servers have to have well-known numbers, so that people can open connections to them and start sending them commands. The official port numbers for each program are given in 'Assigned Numbers', currently RFC 1010.

Note that a connection is actually described by a set of four numbers, the Internet address and the TCP port number at each end. Every datagram has all four of these numbers in it. The Internet addresses are in the IP header, and the TCP port numbers are in the TCP header.

No two connections can have the same set of numbers. However, it is enough for any one number to be different. For example, it is possible for two different users on a machine to be sending files to the same other machine. This could result in connections with parameters as shown below.

```
                 Internet addresses         TCP ports

connection 1  128.6.4.194, 128.6.4.7        1234, 21
connection 2  128.6.4.194, 128.6.4.7        1235, 21
```

Since the same machines are involved, the Internet addresses are the same. Since they are both doing file transfers, one end of the connection involves the well-known port number for FTP. The only item that differs is the port number for the program that the users are running. That single difference is sufficient. Generally, at least one end of the connection asks the network software to assign it a port number that is guaranteed to be unique. Normally, it is the user's end, since the server has to use a well-known number.

Once TCP has opened a connection, we have something that could be a simple wire. All the complex processing is performed by TCP and IP. However we still need some agreement regarding what we send over this connection. In effect, this is an agreement on what set of commands the application will understand, and the format in which they are to be sent.

Generally, what is sent is a combination of commands and data. They use context to differentiate. For example, the `mail` protocol works as follows. `mail` opens a connection to the mail server at the other end. Your program gives it your machine's name, the sender of the message, and the recipients you want it sent to. It then sends a command saying that it is starting the message. At this point, the other end stops treating what it sees as commands, and starts accepting the message. Your end then starts sending the text of the message. At the end of the message, a special mark is sent (a dot in the first column). After that, both ends understand that your program is again sending commands. This is the simplest method, and the one that most applications use.

File transfer is somewhat more complex. The file transfer protocol involves two different connections. It begins like `mail`. The user's program sends commands like 'log me in as this user', 'here is my password', and 'send me the file with this name'. However once the command to send data is sent, a second connection is opened for the data itself. It would be possible to send the data on the same connection, as `mail` does. However file transfers often take a long time. The designers of the file transfer protocol wanted to allow the user to continue issuing commands while the transfer being processed. For example, the user might make an inquiry, or she or he might abort the transfer. Thus the designers used a separate connection for the data and leave the original connection for commands. It is also possible to open command connections to two different computers, and tell them to send a file from one to the other. In that case, the data could not go over the command connection.

Remote terminal connections use a different mechanism. For remote logins, there is only one connection. It normally sends data. When it is necessary to send a command (for examples, to set the terminal type or to change a mode), a special character is used to indicate that the next character is a command. If the user happens to type that special character as data, two of them are sent.

A detailed description of the application protocols is beyond the scope of this document. Two common conventions used by applications are described here. First is the common network representation. TCP/IP is intended to be usable on any computer. Unfortunately, not all computers agree on how data is represented. There are differences in character codes (ASCII vs. EBCDIC), in end-of-line conventions (carriage return, line feed, or a representation using counts), and in whether terminals expect characters to be sent individually or a line-at-a-time. In order to allow computers of different kinds to communicate, each applications protocol defines a standard representation.

Note that TCP and IP do not care about the representation. TCP simply sends octets. However the programs at both ends have to agree on how the octets are to be interpreted. The RFC for each application specifies the standard representation for that application. Normally it is 'net ASCII'. This uses ASCII characters, with end-of-line denoted by a carriage return followed by a line feed.

Second is the convention defining a 'standard terminal' for remote login. This is a half-duplex terminal with echoing happening on the local machine. Most applications also make provisions for the two computers to agree on other representations that they may find more convenient. For example, PDP-10s have 36-bit words. There is a way that two PDP-10s can agree to send a 36-bit binary file. Similarly, two systems that prefer full-duplex terminal conversations can agree on that. However, each application has a standard representation, which every machine must support.

3.1: An SMTP Application Example

An example of a simple mail transfer protocol (SMTP) follows. This is the mail protocol. Assume that a computer named TOPAZ.RUTGERS.EDU wants to send the following message.

```
Date: Sat, 27 Jun 87 13:26:31 EDT
From: hedrick@topaz.rutgers.edu
To: levy@red.rutgers.edu
Subject: meeting

Let's get together Monday at 1pm.
```

The format of the message itself is described by an Internet standard, RFC 822. The standard specifies that the message must be transmitted as net ASCII, i.e. it must be ASCII, with carriage return/linefeed to delimit lines. It also describes the general structure, as a group of header lines, then a blank line, and then the body of the message. Finally, it describes the syntax of the header lines in detail. Generally they consist of a keyword and then a value.

The addressee is indicated as 'LEVY@RED.RUTGERS.EDU'. Initially, addresses were simply 'person @ machine'. However, recent standards are more flexible. There are now provisions for systems to process other systems' mail. This allows automatic forwarding on behalf of computers not connected to the Internet. It can be used to direct mail for a number of systems to one central mail server. There is no requirement that an actual computer by the name of RED.RUTGERS.EDU even exist.

The name servers could be set up so that you mail to department names, and each department's mail is routed automatically to an appropriate computer. It is also possible that the part before the '@' is something other than a user name. It is possible for programs to be set up to process mail. There are also provisions to process mailing lists, and generic names such as 'postmaster' or 'operator'.

The way the message is to be sent to another system is described by RFCs 821 and 974. The program that is going to be doing the sending asks the name server several queries to determine where to route the message. The first query is to find out which machines process mail for the name RED.RUTGERS.EDU. In this case, the server replies that RED.RUTGERS.EDU processes its own mail.

The program then asks for the address of RED.RUTGERS.EDU, which is 128.6.4.2. Then the mail program opens a TCP connection to port 25 on 128.6.4.2. Port 25 is the well-known socket used for receiving mail. Once this connection is established, the mail program starts sending commands. A typical conversation appears below. Each line is labeled whether it is from TOPAZ or RED. Note that TOPAZ initiates the connection.

```
RED     220 RED.RUTGERS.EDU SMTP Service at 29 Jun 87 05:17:18 EDT
TOPAZ   HELO topaz.rutgers.edu
RED     250 RED.RUTGERS.EDU - Hello, TOPAZ.RUTGERS.EDU
TOPAZ   MAIL From:<hedrick@topaz.rutgers.edu>
RED     250 MAIL accepted
TOPAZ   RCPT To:<levy@red.rutgers.edu>
RED     250 Recipient accepted
TOPAZ   DATA
RED     354 Start mail input; end with <CRLF>.<CRLF>
TOPAZ   Date: Sat, 27 Jun 87 13:26:31 EDT
TOPAZ   From: hedrick@topaz.rutgers.edu
TOPAZ   To: levy@red.rutgers.edu
TOPAZ   Subject: meeting
TOPAZ
TOPAZ   Let's get together Monday at 1pm.
TOPAZ   .
RED     250 OK
TOPAZ   QUIT
RED     221 RED.RUTGERS.EDU Service closing transmission channel
```

First, note that the commands all use normal text. This is typical of the Internet standards. Many protocols use standard ASCII commands. This makes it simple to monitor and to diagnose problems. For example, the `mail` program keeps a log of each conversation. If something goes wrong, the log file can be mailed to the postmaster. Since it is normal text, she or he can determine what has occurred. It also allows a human to interact directly with the mail server, for testing.

Some newer protocols are complex enough that this is not practical. The commands would need a syntax requiring a significant parser. Thus there is a tendency for newer protocols to use binary formats. Generally they are structured like C or Pascal record structures.

Second, note that the responses all begin with numbers. This is also typical of Internet protocols. The allowable responses are defined in the protocol. The numbers allow the user program to respond unambiguously. The rest of the response is text, which is normally for use by any human who may be watching or looking at a log. It has no effect on the operation of the programs. Note, however, there is one point at which the protocol uses part of the text of the response.

The commands themselves allow the `mail` program on one end to tell the mail server the information it needs to know in order to deliver the message. In this case, the mail server could get the information by looking at the message itself. But for more complex cases, that would not be safe. Every session must begin with a HELO, which gives the name of the system that initiated the connection. Then the sender and recipients are specified. There can be more than one RCPT command, if there are several recipients.

Finally, the data itself is sent. Note that the text of the message is terminated by a line containing a period. If such a line appears in the message, the period is doubled. After the message is accepted, the sender can send another message, or terminate the session as in the example above.

Generally, there is a pattern to the response numbers. The protocol defines the specific set of responses that can be sent as answers to any given command. However programs that do not want to analyze them in detail can look at the first digit only. Typically, responses that begin with a '2' indicate success. Those that begin with '3' indicate further action is needed, as shown above. Responses of '4' and '5' indicate errors. A '4' is a 'temporary' error, such as a disk filling. The message should be saved, and tried again later. A '5' is a permanent error, such as a non-existent recipient. The message should be returned to the sender with an error message.

For more details about the protocols mentioned in this section, see RFCs 821 and 822 for `mail`, RFC 959 for file transfer, and RFCs 854 and 855 for remote logins. For the well-known port numbers, see the current edition of Assigned Numbers, and possibly RFC 814.

4: UDP and ICMP Protocols

The discussion has included only connections that use TCP thus far. TCP is responsible for breaking-up messages into datagrams, and reassembling them properly. However, in many applications messages will fit into a single datagram. An example is name lookup. When a user attempts to make a connection to another system, she or he will generally specify the system by name, rather than by Internet address. The user's system has to translate that name to an address before it can do anything.

Generally, only a few systems have the database used to translate names to addresses. So the user's system will want to send a query to one of the systems that has the database. This query is going to be very short. It will certainly fit into one datagram, as will the answer. Thus it is not necessary to use TCP. Of course, TCP does more than just break messages up into datagrams. It also makes sure that the data arrives, resending datagrams where necessary. But for a question that fits in a single datagram, we do not need all the complexity of TCP to do this. If we do not get an answer after a few seconds, we can just ask again. For applications like this, there are alternatives to TCP.

The most common alternative is the user datagram protocol (UDP). UDP is designed for applications where you do not need to put sequences of datagrams together. It fits into the system much like TCP. There is a UDP header. The network software puts the UDP header on the front of your data, just as it would put a TCP header on the front of your data. Then UDP sends the data to IP, which adds the IP header, putting the UDP protocol number in the protocol field instead of the TCP protocol number.

However UDP does not do as much as TCP does. It does not split data into multiple datagrams. It does not keep track of what it has sent so it can resend if necessary. UDP provides port numbers, so that several programs can use UDP at once. UDP port numbers are used just like TCP port numbers. There are well-known port numbers for servers that use UDP. Note that the UDP header is shorter than a TCP header. It still has source and destination port numbers, and a checksum. No sequence number is present, since it is not needed. UDP is used by the protocols that process name lookups and a number of similar protocols. See IEN 116, RFC 882, and RFC 883.

Another alternative protocol is the Internet control message protocol (ICMP). ICMP is used for error messages, and other messages intended for the TCP/IP software itself, rather than by any particular user program. For example, if you attempt to connect to a host, your system may get back an ICMP message saying *host unreachable*. ICMP can also be used to find information about the network. See RFC 792 for details of ICMP. ICMP is similar to UDP in that it processes messages that fit in one datagram. However, it is even simpler than UDP. It does not have port numbers in its header. Since all ICMP messages are interpreted by the network software itself, no port numbers are needed to say where a ICMP message is supposed to go..

5: The Domain System: Keeping Track of Names and Information

The network software generally needs a 32-bit Internet address to open a connection or to send a datagram. However, users prefer use computer names rather than numbers. Thus, there is a database that allows the software to look up a name and find the corresponding number.

When the Internet was small, this was easy. Each system had a file that listed all of the other systems, giving both their name and number. There are now too many computers for this approach to be practical. Thus these files have been replaced by a set of name servers that keep track of host names and the corresponding Internet addresses. These servers are somewhat more general, this being just one kind of information stored in the domain system.

Note that a set of interlocking servers is used, rather than a single central one. There are now so many institutions connected to the Internet that it would be impractical for them to notify a central authority whenever they installed or moved a computer. Thus naming authority is delegated to individual institutions. The name servers form a tree, corresponding to institutional structure. The names themselves follow a similar structure. A typical example is the name 'BORAX.LCS.MIT.EDU'. This is a computer at the Laboratory for Computer Science (LCS) at MIT. To find its Internet address, you might have to consult four servers.

First, you would ask a central server, called the root, where the EDU server is. EDU is a server that keeps track of educational institutions. The root server would give you the names and Internet addresses of several servers for EDU. There are several servers at each level, to allow for the possibly that one might be down. You would then ask EDU where the server for MIT is. Again, it would give you names and Internet addresses of several servers for MIT. Generally, not all of those servers would be at MIT, to allow for the possibility of a general power failure at MIT.

Then you would ask MIT where the server for LCS is, and finally you would ask one of the LCS servers about BORAX. The final result would be the Internet address for BORAX.LCS.MIT.EDU. Each of these levels is referred to as a *domain.* The entire name, BORAX.LCS.MIT.EDU, is called a *domain name.* So are the names of the higher-level domains, such as LCS.MIT.EDU, MIT.EDU, and EDU.

You do not have to go do this most of the time. First, the root name servers also are the name servers for the top-level domains such as EDU. Thus, a single query to a root server will get you to MIT. Second, software generally remembers answers that it got before. So once we look up a name at LCS.MIT.EDU, our software remembers where to find servers for LCS.MIT.EDU, MIT.EDU, and EDU. It also remembers the translation of BORAX.LCS.MIT.EDU.

Each of these pieces of information has a *time-to-live* associated with it. Typically this is a few days. After that, the information expires and has to be looked up again. This allows institutions to make changes.

The domain system is not limited to finding Internet addresses. Each domain name is a node in a database. The node can have records that define a number of properties. Examples are Internet address, computer type, and a list of services provided by a computer. A program can ask for a specific piece of information, or all information about a given name. It is possible for a node in the database to be marked as an alias or nickname for another node. It is also possible to use the domain system to store information about users, mailing lists, or other objects.

There is an Internet standard defining the operation of these databases, as well as the protocols used to make queries of them. Every network utility has to be able to make such queries, since this is now the official way to evaluate host names.

Generally, utilities will talk to a server on their own system. This server will take care of contacting the other servers for them. This reduces the amount of code that has to be in each application program.

The domain system is particularly important for processing computer mail. There are entry types to define what computer processes mail for a given name, to specify where an individual is to receive mail, and to define mailing lists.

See RFCs 882, 883, and 973 for specifications of the domain system. RFC 974 defines the use of the domain system in sending mail.

6: Routing

The IP implementation is responsible for getting datagrams to the destination indicated by the destination address. The task of finding how to get a datagram to its destination is referred to as *routing*. In fact, many of the details depend on the particular implementation. However, some general statements may be made.

First, it is necessary to understand the model on which IP is based. IP assumes that a system is attached to some local network. We assume that the system can send datagrams to any other system on its own network. In the case of Ethernet, it simply finds the Ethernet address of the destination system, and puts the datagram out on the Ethernet. The problem comes when a system is asked to send a datagram to a system on a different network. This problem is processed by gateways.

A gateway is a system that connects a network with one or more other networks. Gateways are often normal computers that happen to have more than one network interface. For example, we have a UNIX machine that has two different Ethernet interfaces. Thus, it is connected to networks 128.6.4 and 128.6.3. This machine can act as a gateway between those two networks. The software on that machine must be set up so that it will forward datagrams from one network to the other.

If a machine on network 128.6.4 sends a datagram to the gateway, and the datagram is addressed to a machine on network 128.6.3, the gateway will forward the datagram to the destination. Major communications centers often have gateways that connect a number of different networks. In many cases, special-purpose gateway systems provide better performance or reliability than general-purpose systems acting as gateways. A number of vendors sell such systems.

Routing in IP is based upon the network number of the destination address. Each computer has a table of network numbers. For each network number, a gateway is listed. This is the gateway to use to get to that network. Note that the gateway does not have to connect directly to the network. It just has to be the best place to go to get there.

For example, at Rutgers our interface to NSFnet is at the John von Neuman Supercomputer Center (JvNC). Our connection to JvNC is via a high-speed, serial line connected to a gateway whose address is 128.6.3.12. Systems on net

128.6.3 will list 128.6.3.12 as the gateway for many off-campus networks. However, systems on net 128.6.4 will list 128.6.4.1 as the gateway to those same off-campus networks. Address 128.6.4.1 is the gateway between networks 128.6.4 and 128.6.3, so it is the first step in getting to JvNC.

When a computer wants to send a datagram, it first checks to see if the destination address is on the system's own local network. If so, the datagram can be sent directly. Otherwise, the system expects to find an entry for the network that the destination address is on. The datagram is sent to the gateway listed in that entry. This table can get quite long. For example, the Internet now includes several hundred individual networks. Thus, various strategies have been developed to reduce the size of the routing table. One strategy is to depend upon *default routes*. Often, there is only one gateway out of a network.

This single gateway might connect a local Ethernet to a campus-wide backbone network. In that case, we do not need to have a separate entry for every network in the world. We simply define that gateway as a *default*. When no specific route is found for a datagram, the datagram is sent to the default gateway. A default gateway can be used when there are several gateways on a network. There are provisions for gateways to send a message saying 'I am not the best gateway -- use this one instead'. The message is sent via ICMP. See RFC 792.

Most network software is designed to use these messages to add entries to their routing tables. Suppose network 128.6.4 has two gateways, 128.6.4.59 and 128.6.4.1. Address 128.6.4.59 leads to several other internal Rutgers networks. Address 128.6.4.1 leads indirectly to the NSFnet. Suppose we set 128.6.4.59 as a default gateway, and have no other routing table entries. Now what happens when we need to send a datagram to MIT?

MIT is network 18. Since we have no entry for network 18, the datagram will be sent to the default, 128.6.4.59. As it happens, this gateway is the wrong one. So it will forward the datagram to 128.6.4.1. But it will also send back an error saying in effect that 'To get to network 18, use 128.6.4.1.' Our software will then add an entry to the routing table. Any future datagrams to MIT will then go directly to 128.6.4.1. The error message is sent using the ICMP protocol. The message type is called *ICMP redirect*.

Most IP experts recommend that individual computers should not try to keep track of the entire network. Instead, they should start with default gateways, and let the gateways tell them the routes. However, this does not say how the gateways should find out about the routes. The gateways can not depend on this strategy. They require fairly complete routing tables. For this, a routing protocol is needed.

A routing protocol is a technique for the gateways to find each other, and to keep up-to-date about the best way to get to every network. RFC 1009 contains a review of gateway design and routing. `rip.doc` is an introduction to the subject. It contains some tutorial material, and a detailed description of the most commonly-used routing protocol.

7: Subnets and Broadcasting -- Internet Address Details

Internet addresses are 32-bit numbers, normally written as four octets (in decimal), e.g. 128.6.4.7. There are actually three types of address. The address has to indicate both the network and the host within the network. It was felt that eventually there would be numerous networks. Many of them would be small, but probably 24 bits would be needed to represent all IP networks. It was also felt that some very large networks might need 24 bits to represent all of their hosts. This would seem to lead to 48- bit addresses. But the designers wanted to use 32-bit addresses.

They adopted a compromise. The assumption is that most of the networks will be small. So they set up three ranges of address. Addresses beginning with one to 126 use only the first octet for the network number. The other three octets are available for the host number. Thus 24 bits are available for hosts. These numbers are used for large networks. But there can only be 126 of these very large networks. The ARPAnet is one, and there are a few large commercial networks.

Few normal organizations get one of these 'class A' addresses. For normal large organizations, 'class B' addresses are used. Class B addresses use the first two octets for the network number. Thus, network numbers are 128.1 through 191.254. We avoid zero and 255, for reasons described below. We also avoid addresses beginning with 127, because that is used by some systems for special purposes. The last two octets are available for host addresses, giving 16 bits of host address. This allows for 64,516 computers, which should be enough for most organizations. It is possible to get more than one class B address, if necessary.

Finally, class C addresses use three octets, in the range 192.1.1 to 223.254.254. These allow only 254 hosts on each network, but there can be many of these networks. Addresses above 223 are reserved for future use, as class D and E, which are currently not defined.

Many large organizations find it convenient to divide their network number into *subnets*. For example, Rutgers has been assigned a class B address, 128.6. We find it convenient to use the third octet of the address to indicate which Ethernet a host is on. This division has no significance outside of Rutgers. A computer at another institution would treat all datagrams addressed to 128.6 the same way. They would not look at the third octet of the address.

Thus, computers outside Rutgers would not have different routes for 128.6.4 or 128.6.5. But inside Rutgers, we treat 128.6.4 and 128.6.5 as separate networks. In effect, gateways inside Rutgers have separate entries for each Rutgers subnet, whereas gateways outside Rutgers have but one entry for 128.6. Note that we could do the same by using a separate class C address for each Ethernet. As far as Rutgers is concerned, it would be just as convenient for us to have a number of class C addresses. However using class C addresses would be inconvenient for the rest of the world.

Every institution that wanted to talk to us would have to have a separate entry for each one of our networks. If every institution did this, there would be far too many networks for any reasonable gateway to monitor. By subdividing a class B network, we hide our internal structure from everyone else, and save them the trouble. This subnet strategy requires special provisions in the network software. It is described in RFC 950.

Zero and 255 have special meanings. Zero is reserved for machines that do not know their address. In certain circumstances, it is possible for a machine not to know the number of the network it is on, or even its own host address. For example, 0.0.0.23 would be a machine that knew it was host number 23, but did not know on what network.

Address 255 is used for *broadcast*. A broadcast is a message that you want every system on the network to see. Broadcasts are used in some situations where you do not know who to talk to. For example, suppose you need to look up a host name and get its Internet address. Sometimes you do not know the address of the nearest name server. In that case, you might send the request as a broadcast. There are also cases where a number of systems are interested in information. It is then less expensive to send a single broadcast than to send datagrams individually to each host that is interested in the information.

In order to send a broadcast, you use an address that is made by using your network address, with all ones (1's) in the part of the address used for the host number. For example, if you are on network 128.6.4, you would use 128.6.4.255 for broadcasts. How this is actually implemented depends upon the medium. It is not possible to send broadcasts on the ARPAnet, or on point-to-point lines. However, it is possible on an Ethernet. If you use an Ethernet address with all ones (1's), every machine on the Ethernet is supposed to look at that datagram.

Although the official broadcast address for network 128.6.4 is now 128.6.4.255, there are some other addresses that may be treated as broadcasts by certain implementations. For convenience, the standard also allows 255.255.255.255 to be used. This refers to all hosts on the local network. It is often simpler to use 255.255.255.255 instead of finding the network number for the local network and forming a broadcast address such as 128.6.4.255. In addition, certain older implementations may use zero instead of 255 to form the broadcast address. Such implementations would use 128.6.4.0 instead of 128.6.4.255 as the broadcast address on network 128.6.4.

Finally, certain older implementations may not understand about subnets. Thus, they consider the network number to be 128.6. In that case, they will assume a broadcast address of 128.6.255.255 or 128.6.0.0. Until support for broadcasts is implemented properly, it can be a somewhat dangerous feature to use.

Because zero and 255 are used for unknown and broadcast addresses, normal hosts should never be given addresses containing zero or 255. Addresses should never begin with zero, 127, or any number above 223.

8: Datagram Fragmentation and Reassembly

TCP/IP is designed for use with many kinds of networks. Unfortunately, network designers do not agree on how large packets can be. Ethernet packets can be 1,500 octets long. ARPAnet packets have a maximum of approximately 1,000 octets. Some very fast networks have much larger packet sizes.

IP cannot simply settle on the smallest possible size. This would cause serious performance problems. When transferring large files, large packets are far more efficient than small ones. So we want to be able to use the largest packet size possible. But we also want to be able to communicate with networks using small packet limits.

There are two provisions for this. First, TCP has the ability to 'negotiate' datagram size. When a TCP connection first opens, both ends can send the maximum datagram size they process. The smaller of these limits is used for the rest of the connection. This allows two implementations that can process large datagrams to use them, but also lets them talk to implementations that cannot process them. However, this does not completely solve the problem. The most serious problem is that the two ends do not necessarily know about all of the steps in between.

For example, when sending data between Rutgers and Berkeley, it is likely that both computers will be on Ethernets. Thus they will both be prepared to process 1,500-octet datagrams. However the connection will at some point end up going over the ARPAnet. It can not process packets of that size. For this reason, there are provisions to split datagrams up into pieces. This process is referred to as *fragmentation.*

The IP header contains fields indicating that a datagram has been split, and enough information to let the pieces be put back together. If a gateway connects an Ethernet to the ARPAnet, it must be prepared to take 1,500-octet Ethernet packets and split them into pieces that will fit on the ARPAnet. Furthermore, every host implementation of TCP/IP must be prepared to accept pieces and put them back together. This is referred to as *reassembly.*

TCP/IP implementations differ in the approach they take to deciding on datagram size. It is fairly common for implementations to use 576-byte datagrams whenever they can not verify that the entire path is able to process larger packets. This rather conservative strategy is used because of the number of implementations with bugs in the code to reassemble fragments. Implementors often try to avoid ever having fragmentation occur. Different implementors take different approaches to deciding when it is safe to use large datagrams. Some use them only for the local network. Others will use them for any network on the same campus. A 'safe' size is 576 bytes, which every implementation must support.

9: ARP -- Ethernet Encapsulation

This discussion details how to determine which Ethernet address to use when you want to talk to a given Internet address. In fact, there is a separate protocol for this, called the address resolution protocol (ARP).

ARP is not an IP protocol. That is, the ARP datagrams do not have IP headers. Suppose you are on system 128.6.4.194 and you want to connect to system 128.6.4.7. Your system will first verify that 128.6.4.7 is on the same network, so it can talk directly via Ethernet. Then it will look up 128.6.4.7 in its ARP table, to see if it already knows the Ethernet address. If so, it will add an Ethernet header, and send the packet.

But suppose this system is not in the ARP table. There is no way to send the packet, because you need the Ethernet address. So it uses the ARP protocol to send an ARP request. Essentially an ARP request says 'I need the Ethernet address for 128.6.4.7.' Every system listens to ARP requests. When a system sees an ARP request for itself, it is required to respond. So 128.6.4.7 will see the request, and will respond with an ARP reply saying in effect '128.6.4.7 is 8:0:20:1:56:34.'

Recall that Ethernet addresses are 48 bits. This is six octets. Ethernet addresses are conventionally shown in hex, using the punctuation shown. Your system will save this information in its ARP table, so future packets will go directly. Most systems treat the ARP table as a cache, and clear entries in it if they have not been used in a certain period of time.

Note that ARP requests must be sent as broadcasts. There is no way that an ARP request can be sent directly to the right system. After all, the whole reason for sending an ARP request is that you do not know the Ethernet address. So an Ethernet address of all ones (1's) is used, i.e. ff:ff:ff:ff:ff:ff. By convention, every machine on the Ethernet is required to pay attention to packets with this as an address. So every system sees every ARP requests. They all look to see whether the request is for their own address. If so, they respond. If not, they could just ignore it. Some hosts will use ARP requests to update their knowledge about other hosts on the network, even if the request is not for them. Note that packets whose IP address indicates broadcast (e.g. 255.255.255.255 or 128.6.4.255) are also sent with an Ethernet address that is all ones (1's).

10: Getting More Information

The references for more information contained in the following paragraphs include some of the many documents describing the major protocols. Internet standards are called request for comments (RFCs). A proposed standard is initially issued as a proposal, and given an RFC number. When it is finally accepted, it is added to 'Official Internet Protocols', but it is still referred to by the RFC number.

We have also included two IENs, which used to be a separate classification for more informal documents. This classification no longer exists. RFCs are now used for all official Internet documents, and a mailing list is used for more informal reports. The convention is that whenever an RFC is revised, the revised version gets a new number. This is fine for most purposes, but it causes problems with two documents, *Assigned Numbers* and *Official Internet Protocols*. These documents are being revised all the time, so the RFC number keeps changing. You will have to look in `rfc-index.txt` to find the number of the latest edition. See RFC 791 which describes IP.

RFC 1009 is also useful. It is a specification for gateways to be used by NSFnet. As such, it contains an overview of a lot of the TCP/IP technology. Read the description of at least one of the application protocols. `mail` is a good one, RFCs 821 and 822. TCP 793 is of course a very basic specification. However, the specification is fairly complex.

10.1: Helpful General Documents

A number of helpful documents are described below.

`rfc-index`	list of all RFCs
`rfc1012`	somewhat fuller list of all RFCs
`rfc1011`	Official Protocols. It is useful to scan this to see which tasks for which the protocols have been built. This defines which RFCs are actual standards and which are requests for comments.
`rfc1010`	Assigned Numbers. If you are working with TCP/IP, you will probably want a hardcopy of this as a reference. It lists all the officially defined well-known ports and other topics.
`rfc1009`	NSFnet gateway specifications. A good overview of IP routing and gateway technology.
`rfc1001/2`	netBIOS: networking for PCs
`rfc973`	update on domains
`rfc959`	FTP (file transfer)
`rfc950`	subnets
`rfc937`	POP2: protocol for reading mail on PCs
`rfc894`	how IP is to be put on Ethernet. See also rfc825.
`rfc882/3`	domains, the database used to go from hostnames to Internet address and back, also used to process UUCP. See also `rfc973`.
`rfc854/5`	`telnet`, a protocol for remote logins
`rfc826`	ARP, a protocol for finding Ethernet addresses
`rfc821/2`	`mail`
`rfc814`	names and ports, general concepts behind well-known ports

`rfc793`	TCP
`rfc792`	ICMP
`rfc791`	IP
`rfc768`	UDP
`rip.doc`	details of the most commonly-used routing protocol
`ien-116`	old name server, needed by several kinds of systems
`ien-48`	the Catenet model, general description of the philosophy behind TCP/IP

10.2: Helpful Specialized Documents

The following documents are somewhat more specialized.

`rfc813`	window and acknowledgement strategies in TCP
`rfc815`	datagram reassembly techniques
`rfc816`	fault isolation and resolution techniques
`rfc817`	modularity and efficiency in implementation
`rfc879`	the maximum segment size option in TCP
`rfc896`	congestion control
`rfc827,888,904,975,985`	EGP and related issues

The most important RFCs have been collected into a three-volume set, the *DDN Protocol Handbook*. It is available from the DDN Network Information Center, SRI International, 333 Ravenswood Avenue, Menlo Park, California 94025, telephone (800) 235-3155. You should be able to get them via anonymous FTP from *sri-nic.arpa*. File names are shown below.

```
RFCs:   rfc:rfc-index.txt
        rfc:rfcxxx.txt

IENs:   ien:ien-index.txt
        ien:ien-xxx.txt
```

`rip.doc` is available by anonymous FTP from *topaz.rutgers.edu*, as `/pub/tcp-ip-docs/rip.doc`.

Sites with access to UUCP but not FTP may be able to retrieve them via UUCP from UUCP host rutgers. The file names would be as shown below.

```
RFCs:    /topaz/pub/pub/tcp-ip-docs/rfc-index.txt
         /topaz/pub/pub/tcp-ip-docs/rfcxxx.txt

IENs:    /topaz/pub/pub/tcp-ip-docs/ien-index.txt
         /topaz/pub/pub/tcp-ip-docs/ien-xxx.txt
         /topaz/pub/pub/tcp-ip-docs/rip.doc
```

Note that SRI-NIC has the entire set of RFCs and IENs, but `rutgers` and `topaz` have only those specifically mentioned above.

5

QUESTIONS, ANSWERS, HINTS, AND TIPS

5

QUESTIONS, ANSWERS, HINTS, AND TIPS

Q&A, and Tip of the Month

Hints & Tips #5

This is the fifth in a continuing series of this column which I have created for two purposes.[7] First, some questions are asked regularly on the AnswerLine. I feel everyone can benefit from distributing discussions of these problems as widely as possible. Second, a large and constantly growing body of information, hints, and tips are not documented anywhere.

I will collect and distribute these information nuggets in this continuing column so that we can all learn from them. I will cover unusual topics, but this column should not be used as an alternative to contacting your support center or using the AnswerLine.

If you have a question that you would like answered in this column, please mail your question to 'Software Technical Bulletins' at Sun Microsystems, Inc., 2550 Garcia Avenue, M/S 2-34, Mountain View, CA 94043. You can also send in your question by electronic mail to *sun!stb-editor*. U. S. customers can call Sun Customer Software Services AnswerLine at **800 USA-4-SUN** for technical questions on this column or any other article in this bulletin. I look forward to hearing from you!

Tip of the Month (TOM)

This month is an extended Tip of the Month that includes a discussion of using workstations with different monitors in ways that make such differences transparent to the user.

Determining Your Monitor Type

One common example of this problem is when some of your machines have a standard monochrome monitor and others have a high-resolution or color monitor. Most people want to use different versions of the `.suntools` and `.defaults` files on a high-resolution screen so that they can use a larger font or change window sizes.

[7] This continuing column is submitted by Chuq Von Rospach, Customer Software Services.

To do this, you need to know which type screen you are using. The program shown on the next page, called `fbres`, will look at your frame buffer and tell you what you are using.

```
/*
 *      fbres.c
 *
 *      Prints out the resolution of the fb pixrect in a
 *      form useful for shell scripts.
 *
 *      Usage: fpres [fb-name]
 */

#include <stdio.h>
#include <pixrect/pixrect_hs.h>

main(argc,argv)
int     argc;
char    *argv[];
{
    Pixrect *fbpr;
    char    *fbname;

    if (argc > 1)
        fbname = argv[1];
    else
        fbname = "/dev/fb";

    if ((fbpr = pr_open(fbname)) == NULL) {
        fprintf(stderr,"Couldn't open %s pixrect\n",fbname);
        exit(1);
    }
    printf("%dx%d\n",fbpr->pr_size.x,fbpr->pr_size.y);
}
```

Appropriate Defaults for Each Monitor Type

With SunOS release 3.4, it is now possible to set the environment variable `DEFAULTS_FILE` to tell `suntools` what 'defaults' file to use. With this feature and a `csh` alias, you can automatically customize your `suntools` environment to use the defaults appropriate to the machine you are on. The following fragment of `csh` code is from the `.login` file, and is executed when you log in.

```
if (`tty` == /dev/console) then
if (`fbres` == "1600x1280") then
    setenv DEFAULTS_FILE  /.defaults.hires
    alias st 'cd;exec suntools -F -s .suntools.hires'
else
    setenv DEFAULTS_FILE  /.defaults
    alias st 'cd;exec suntools -F'
endif
input_from_defaults
endif
```

Once this is done, executing `st` will start `suntools` with the appropriate defaults and screen layout.

Sun4 Binary Compatibility

Binary compatibility is going to become an area of interest to Sun customers with the introduction of the Sun4 line of computers. Previously, if you needed a program to run under both Sun2 and Sun3 machines, you could compile it on a Sun2. However, the Sun4 uses a new instruction set so this type of arrangement will no longer work. If you have a home directory that is NFS-mounted onto both a Sun3 and a Sun4, you will find that your private programs will fail on one machine or another unless you set up your account to take the machine differences into consideration.

One solution that is being used at this time is to set up the private `/bin` directory as shown below.

```
/homedir
    /bin        -- shell scripts and non-compiled programs
        /sun3   -- Sun3 binaries
        /sun4   -- Sun4 binaries
```

All programs that are architecture-independent (shell scripts, `awk` scripts, and the like) are placed in the `/bin` directory as always. All of the programs that need to be compiled, however, need to be compiled *for each architecture* and placed in a separate subdirectory. You can then use the `arch` command in your `.login` file to modify your $PATH at log in time to include only those directories that are useful to the machine to which you are logged on. An example is shown below.

```
set path=(. $HOME/bin $HOME/bin/`arch` /usr/ucb /bin /usr/bin)
```

Now, when you log in, your account will customize itself to your architecture, screen type, and keep you from having to remember architecture-specific details, or worrying about which commands work on a given machine.

6

THE HACKERS' CORNER

6

THE HACKERS' CORNER

C Calling NeWS

The Hackers' Corner: Calling NeWS from C

There are times when it might be helpful to call NeWS from a C program and return a text-string that is entered in real time by the user.

Professional Interest

The script or code contained in this article may be of interest to professionals, enthusiasts, or anyone having the time to key the script or code onto their system.

Also, please consult your local shell script or programming expert regarding any script or code problems. The script or code is not offered as a supported Sun product, but as an item of interest to enthusiasts wanting to try out something for themselves.

Using The Archive

The `shar` archive in this article can be built by following the steps shown below.

1. Save the archive into a file named, for example, `news.archive`, in an otherwise empty directory.

2. Pass the file through *sh(1)* by using the command shown below.

   ```
   % sh < news.archive
   ```

3. Set any parameters in the makefile.

4. Use the command shown below.

   ```
   % make
   ```

5. Run the program `getstr` from a machine that can connect to a NeWS server.

6. Type text at the line marked 'String' after clicking the mouse in the box.

7. Select 'Send String' from the menu.

The string will now be sent to the `stdout` of the location that you started the program.

The Archive: Calling NeWS from C

The `shar` archive appears on the following pages.

```
# This is a shell archive.  Remove anything before this line,
# then unpack it by saving it in a file and typing 'sh file'.
#
# Contents:  Makefile getstr.c getstr.cps

echo x - Makefile
sed 's/^@//' > "Makefile" <<'@//E*O*F Makefile//'
NEWSLIB= /usr/NeWS/lib

ALL: getstr getstr.h

getstr.h: getstr.cps
    cps getstr.cps

getstr.o: getstr.h
    cc -c getstr.c

getstr: getstr.o
    cc -o getstr getstr.o $(NEWSLIB)/libcps.a

clean:
    rm -f getstr.o getstr.h getstr.h.BAK
@//E*O*F Makefile//
chmod u=rwx,g=rx,o=rx Makefile

echo x - getstr.c
sed 's/^@//' > "getstr.c" <<'@//E*O*F getstr.c//'
/*  A very simple NeWS client */

#ifndef lint
static  char sccsid[] = "%M% %I% %E%";
#endif

#include <stdio.h>
#include "getstr.h"

main()
{
    float fillgray = .75;
    char h_string[100];

    if (ps_open_PostScript() == 0 ) {
    printf(stderr,"Cannot connect to NeWS server");
    exit(1);
    }

    initialize();

    while ( !ferror(PostScriptInput) ) {
    if      (get_gray(&fillgray))   call_paint_client();
    else if (get_paint_client())    paint_client(fillgray);
    else if (get_str(h_string)) printf("This is it: %s\n", h_string);
    else if (get_done())        {printf ("Done!\n");break;}
    else if (feof(PostScriptInput)) break;
    else                {printf ("Strange Stuff!\n");break;}
    }

    ps_close_PostScript();
}
@//E*O*F getstr.c//
chmod u=rw,g=rw,o=r getstr.c

echo x - getstr.cps
sed 's/^@//' > "getstr.cps" <<'@//E*O*F getstr.cps//'
%
% %M% %I% %E%
%

#define SET_GRAY_TAG        1
#define PAINT_CLIENT_TAG    2
#define DONE_TAG        3
#define GET_STR_TAG     4

cdef initialize()
    % see if the items stuff is around...
    systemdict /Item known not { (NeWS/liteitem.ps) run } if
    /hold_string (<Current String comes here>) def
    /notify? true def
    /notify {
        notify? {(Notify: Value=%) [ItemValue] /printf messages send
         userdict /hold_string ItemValue put } if
    } def

    /createitems {
       /items 50 dict dup begin
          /nameitem (String:) () /Right /notify can 220 0
          /new TextItem send 20 260 /move 3 index send def

          /messages /panel_text hold_string /Right {} can 500 0
          /new MessageItem send dup begin
             /ItemFrame 1 def
             /ItemBorder 4 def
             end 20 290 /move 3 index send def
          end def
          /messages items /messages get def
       } def

    /win framebuffer /new DefaultWindow send def
    {   /FrameLabel (GetStr) def
    /IconLabel (GetStr) def
    /PaintClient {PAINT_CLIENT_TAG tagprint} def
    /ClientMenu [
        (Send String) {  hold_string GET_STR_TAG tagprint typedprint}
        (White)     {  1 SET_GRAY_TAG tagprint typedprint}
        (Lite)      {.75 SET_GRAY_TAG tagprint typedprint}
        (Gray)      {.50 SET_GRAY_TAG tagprint typedprint}
        (Dark)      {.25 SET_GRAY_TAG tagprint typedprint}
        (Black)     {  0 SET_GRAY_TAG tagprint typedprint}
    ] /new DefaultMenu send def
    } win send
    200 200 700 350 /reshape win send
    /can win /ClientCanvas get def

    % Create Items
    createitems
    /reshapefromuser win send
    /map win send
    /itemmgr items forkitems def

cdef get_gray(float fillgray) => SET_GRAY_TAG (fillgray)
cdef get_str(string h_string) => GET_STR_TAG (h_string)
cdef get_paint_client() => PAINT_CLIENT_TAG()
cdef get_done() => DONE_TAG()

cdef call_paint_client()
    /paintclient win send
cdef paint_client(float fillgray)
    {ClientCanvas setcanvas fillgray fillcanvas items paintitems} win send


@//E*O*F getstr.cps//
chmod u=rw,g=rw,o=r getstr.cps

exit 0
```

7

CUSTOMER DISTRIBUTED BUGSLIST

CUSTOMER DISTRIBUTED BUGSLIST

Customer Distributed BugsList

Organization

The Customer Distributed BugsList (CDB) is divided into two parts: open bugs in SunOS and open bugs in other Sun software products. Both of these sections are sorted by topic and then by subtopic.

Individual Entries

Each entry in the Customer Distributed BugsList consists of the Reference Number of the bug, a one-line synopsis of the bug, the release(s), a description of the bug, and a workaround for the bug where it is available. The Reference Number is an identification tag for the bug. Use this number when asking your support center about a particular entry. The release(s) is that in which the bug was reported. The description of the bug includes the problem, succinct examples of the problem where available, and the configuration where applicable.

Content

The entries were extracted from our database on 7/16/87. The CDB includes SunOS 3.4 and earlier releases. Some releases include alpha, beta, and pilot phases. Entries showing the releases alpha, beta, and pilot are still open at the time of first customer ship of the release. Refer to the table titled 'Current Sun Products and Release Levels' include in section 1, 'Notes and Comments', of this issue for more information about current releases.

All bugs included in the CDB have been evaluated by our engineering staff. Of these some bugs were eliminated based on the following criteria:

1. The entry was not considered to be a bug, but an in-house request for enhancement.

2. The bugs referred to in-house situations only.

SunOS

Compilers
Assembler

Reference Number: 1003562
Synopsis: Missing Diagnostic - table overflow for switch statements
Release: 3.2

Description:
By default, the compiler/assembler generate 16 bit jump tables for switch statements. If the switch statement is so big that it doesn't fit in 16 bits, bad values are placed in the jump table, causing incorrect code. Some combination of the compiler and/or assembler should warn that this has occurred.
As of 3.2, the compiler supports the -J flag, which causes the compiler/assembler to generate correct code, but the compiler/assembler doesn't inform you that the -J flag is necessary.

Work around:
The -J flag generates correct code.

Reference Number: 1003597
Synopsis: "as" core dumps when asked to divide by zero
Release: 4.0, 3.4, 3.3, 3.2

Description:
If you have an expression in an assembler-language program that attempts to divide by zero, instead of catching this and printing an error message, it continues, does the division, and core dumps. This was discovered through use of an incorrect Makefile. The Makefile did not run the assembly source through the C preprocessor, and somehow this caused it to see code that did a divide by zero.

Reference Number: 1003869
Synopsis: using -mc68010 and -R on command line can make /bin/as core dump
Release: 3.2

Description:
If the user executes /bin/as using both the -mc68010 and -R options, the result is a core dump. The sample assembly file is one line long:

```
largo% more z.s
tstb    0xe00008

largo% /bin/as -mc68010 -R z.s
Segmentation fault (core dumped)
```

Using either -mc68010 or -R alone will not cause this error.

Reference Number: 1003951
Synopsis: as broke on 68881 branch codes
Release: 3.2

Description:
If the pseudo branch's length differs from the real length, /bin/as core dumps on jumping to absolute address and phase error

The following one-line program causes /bin/as to core dump:

```
fjeq   30
```

The following three-line program causes /bin/as to have a phase error:

```
fbnel  L33
.skip  380
L33:
```

C Compiler

Reference Number: 1002757
Synopsis: C compiler core dumps on simple syntax error
Release: 3.2

Description:
The C compiler core dumps on the following simple C program which has a syntax error (attempting to print a member of a two dimensional array by only using the row index):

```
char foo[3][256] = { "one"; "two"; "three"; };
```

```
main()
{
int i;
for (i = 0; i< 3; i++)
printf("%s\n", foo[i]);
}
```

Reference Number: 1002825
Synopsis: c compiler removes .o files unnecessarily
Release: 3.2

Description:
The C compiler should not remove .o files.

Reference Number: 1003017
Synopsis: "cc -a" (tcov profiling) bug
Release: 3.2, 3.0

Description:
"cc -a" gives the following error message when C code contains a local definition for a pointer to an array of doubles:

```
is_func is confused 91 [ [
```

For example, the program foo produces the aforementioned error message when compiled with "cc -a -c":

```
foo()
{
        double (*bar)[5];
}
```

Work around:
Use typedefs. For example:

```
typedef double array[5];
void
foo()
{
array *bar;
...
}
```

Reference Number: 1004497
Synopsis: "cc -a" causes core dump of core dump in C compiler
Release: 3.2

Description:
When "cc -a" is attempted on some files, the result is a segmentation fault and core dump in bb_count, one of the phases of cc where code is inserted to generate information used by TCOV.
Work around:
This is a result of errors in "bb_count" (one of the phases of cc where code is inserted to generate information used by TCOV). "bb_count" is is called after "cpp"(the C pre-processor) and before "ccom"(scanner- parser-intermediate code generator of the C compiler). When "cc -a" is attempted on some files, cc complains about syntax errors in the source which are unwarranted.

Reference Number: 1004498
Synopsis: "cc -a" complains of syntax error in source
Release: 3.2

Description:
When "cc -a" is attempted on some files, cc gives unwarranted syntax errors for the source.

This is a result of errors in "bb_count"(one of the phases of cc where code is inserted to generate information used by TCOV). "bb_count" is is called after "cpp"(the C pre-processor) and before "ccom"(scanner- parser-intermediate code generator of the C compiler).

"bb_count" identifies basic blocks. So when it sees the {....}, it believes that it has a new block. This is problematic if there are struct definitions, macros in the declarations section. Seeing the {....} "bb_count" inserts code that "ccom" complains about.
Work around:
This is a result of errors in "bb_count" (one of the phases of cc where code is inserted to generate information used by TCOV). "bb_count" is is called after "cpp"(the C pre-processor) and before "ccom"(scanner- parser-intermediate code generator of the C compiler). One can work around this, by eliminating {...} constructs in the declaration section.

Reference Number: 1004157
Synopsis: brk and sbrk are incompatible in gprof-profiling
Release: 3.2

Description:

The call to brk(2) in the following program seems to be ignored by sbrk when the program is compiled with the -pg option.

```
bach% cat tst.c
main()
{
printf("%x\n", sbrk(0));
printf("%x\n", brk(0x100000));
printf("%x\n", sbrk(0));
printf("%x\n", sbrk(0x50000));
printf("%x\n", sbrk(0));
}
bach% make
cc tst.c -o tst1
cc -pg tst.c -o tst2
bach% tst1
22e04
0
100000
100000
150000
bach% tst2
2bdd0
0
2ddd0
2ddd0
7ddd0
bach%
```

The problem here is that gmcrt0.o has its own version of brk(), which is global, and its own version of curbrk, which is not. The brk() call updates its local curbrk, and thus fails to communicate with sbrk().

Reference Number: 1004403
Synopsis: bit field assignment and comparison fails
Release: 3.2

Description:

The following code shows an instance in which the C compiler does not properly represent the result of field assignments. The code produces the line "fails", rather than "succeeds":

```
struct {
    int b : 8;
} s;
main()
{
    if ((s.b = 0xff01) != 1) {
        printf("fails\n");
    } else {
        printf("succeeds\n");
    }
}
```

Work around:

Do the assignment on a line before the test.

Reference Number: 1004502
Synopsis: sscanf %c & %[conversions corrupt return address.
Release: 3.2

Description:

The following program expects arguments that consist of <number> <whitespace> <rest-of-line> and use sscanf() to split the argument. When the argument is erroneous (eg. starting with a non-digit character) a 'bus error' occurs on the return to main.

```
    main( argc, argv)
    char** argv;
{
    while (*(++argv)) {
        int n0, n1;
        char  s[100];
        *s = '\0';
        n0 = sscanf( *argv, "%d %100c", &n1, s);
        printf( "%d %d '%s'\n", n0, n1, s);
    }
}
```

Reference Number: 1004517
Synopsis: modulus operator returns an incorrect value
Release: 3.2

Description:

The following C program shows that mod by 1 with assignment (x %= 1) is not compiled correctly.

```
main()
{

int I = 5;
int J = 5;
int K = 1;

I %= 1;
J %= K;

printf("%d %d\n", I,J);
}

% test
5 0
```

Reference Number: 1004564
Synopsis: ccom fails to convert to unsigned in comparison
Release: 3.2

Description:

In the program below, the compiler fails to change the comparison to an unsigned one and thus the program incorrectly produces the following line:

```
x (126) > y (128)
```

```
main()
{
unsigned char x;
unsigned char y;
x = 126;
y = 128;
if ((int) x > (int) y) {
        printf("x (%d) > y (%d)\n", x, y);
} else {
        printf("x (%d) <= y (%d)\n", x, y);
}
}
```

Reference Number: 1005006
Synopsis: C Compiler catching signals when it shouldn't
Release: 3.2

Description:
The C compiler sets up signal handling when started up in background mode from a shell script. For example, the customer uses a Bourne shell script that catches signals 1, 2, and 3 to compile a C program in background mode. After the program compiles, the compiler echoes its return code ('$?'). If the customer runs the script to completion, the compiler echoes a zero. However, if the customer hits control-C (^C), the compiler echoes '1' showing a failure to complete the compilation even though an 'abort' should not effect a compilation run in background mode.

Debuggers

Reference Number: 1001671
Synopsis: 'dbx' routine 'eval.c' causes 'dbx' to panic on a 'popsmall'
Release: 3.0

Description:
In the 'dbx' routine 'eval.c' if a 'popsmall' is a condition 'dbx' panics and exits. This, however, is a legal condition.

Reference Number: 1001693
Synopsis: 3.2Beta 'dbx' core dumps with large files.
Release: 3.2beta

Description:
Using 'dbx' with a large file (38000 lines) causes 'dbx' to core dump.

Reference Number: 1001696
Synopsis: "dbx" doesn't understand Sun a.out files
Release: 3.2alpha

Description:
If you run 'dbx' against an executable image and a core file, it doesn't understand that the text addresses run from N_TXTADDR(header) to N_TXTADDR(header) + core.c_tsize
It thinks that the data segment starts at core.c_tsize.

Reference Number: 1001702
Synopsis: dbxtool uses wrong filename after "up"/"down" and won't set breakpoint
Release: 3.2alpha, 3.2

Description:

After stopping at a breakpoint in dbxtool, the user uses "up" one or more times, selects a line, and attempts to set a breakpoint using the "stop at" button. The "expand" feature for this button generated:

```
stop at "./../src/gp1_prims.c":80
```

This causes 'dbx' to give the following error message:
file "./../src/gp1_prims.c" was not compiled with the "-g" option
When the user continues to step through the current subroutine, and upon returning to the function in which the user wished to breakpoint (ie. same level that "up" took the user to previously).
The "stop at" button then generated:
stop at "../src/gp1_prims.c":80
and 'dbxtool' set the breakpoint.

Work around:

Retype the line 'dbxtool' printed as the expansion without the path in front of filename.c.

For example, 'dbxtool' prints out as follows:

```
(dbxtool) stop at "./../src/gp1_prims.c":80
file "./../src/gp1_prims.c" was not compiled with the "-g" option
(dbxtool)
```

Retype the line as follows:

```
(dbxtool) stop at "gp1_prims.c":80
```

Reference Number: 1001710
Synopsis: 'dbx' Prints multiple lines
Release: 3.2beta

Description:

If you have multiple "stop at" or "when at" statements and the trace statement is within the range of these "stop at" or "when at" statements then the run will print that number of trace statments.
eg: 4 "stop at" statments, 4 "trace" statements.

Reference Number: 1004308
Synopsis: adb -w byte swaps char string writes to executables
Release: 3.2

Description:

If you write a word in an executable using a 4 byte character string as in the example that follows, the result is byte swapped.

Note: The example shows that the W instruction in adb now works intuitively, but the C compiler still behaves in VAX fashion when given multiple-character constants by giving a result of "badco".

```
zeus>> more t.c
char str[]="hello perry";
main()
{
printf("%s\n,str);
}
zeus>> cc t.c
zeus>> adb -w a.out -
str?s
_str:
_str:       hello perry
str?W 'abcd'
_str:       0x68656c6c    =    0x62616463
str?s
_str:
_str:       badco perry
$q
zeus>>
```

The result should be "abcdo" rather than "badco".

Work around:

Use Hex.

Reference Number: 1004996
Synopsis: 'dbx' shows segmentation violation while stepi'ing
Release: 3.4beta, 3.2

Description:

In 'dbx' a segmentation violation occurs when using the 'stepi' command to skip past a 68881 multiplication instruction.

Work around:

Use 'adb' for this case.

FORTRAN Compiler

Reference Number: 1000214
Synopsis: ioinit(3F) does not work as documented.
Release: 2.0beta, 1.4, 1.1

Description:
The documentation states that 'ioinit(3F) initializes several global parameters in the f77 I/O system, and attaches externally defined files to logical units at run time'. This does not work as stated for an application that attempts to use ioinit(3F) to set the filename for the duration of the program run for particular logical unit numbers.
Work around:
Call 'ioinit' before each 'open'.

Reference Number: 1000215
Synopsis: ioinit(3F) requires redundant loading of -lI77
Release: 2.0beta, 1.4, 1.1

Description:
The FORTRAN statement 'ioint(3F)' requires a explicit specification of -lI77 to resolve all references.
Work around:
Explicitly specify -lI77 in the compilation or in the loading command line as in the following:

```
f77 foo.f -lI77
```

Reference Number: 1000271
Synopsis: calling ioinit as per doc. causes link error
Release: 3.2alpha, 3.0

Description:
When you call ioinit as documented on the Fortran page ioinit(3f). You get linker undefined errors. Here are the errors:

```
Undefined:
_s_cmp
_i_len
_lnblnk_
_i_indx
```

Note: these are all routines from libF77a.
Work around:

```
cd to work directory.
ar x /usr/lib/libI77.a ioinit.o
        now ioinit is in your current working directory
f77 *.f ioinit.o
```

You will no longer have these undefined errors.

Reference Number: 1002618
Synopsis: the -C option can cause f77pass1 to core dump
Release: 3.2, 3.2alpha, 3.0

Description:
Fortran77 does not handle reads from internal files correctly: f77pass1 core dumps upon compilation.

Reference Number: 1002628
Synopsis: 'dbx' prints wrong value for real array cells
Release: 3.2alpha, 3.0

Description:
When the following Fortran program is compiled with the -g option for the debugger, the values of the array 's' are printed incorrectly.

```
program total
parameter(np=158,nps=128)
dimension s(np)

data (s(i),i=1,np)/np*0/

open(unit=1,file='s.dat')
read(1,*)(s(i),i=1,nps)
close(1)

print *,s
end
```

The compiler places variables in either the initialized data area or the un-initialized data area depending on whether they have been initialized. All variables in the un-initialized data area have an initial value of 0. When an array is initialized to all 0's, the compiler realizes it can place the array in the uninitialized data area (this makes the object file smaller). However, the debug information it generates says the array is in the initialized data area so 'dbx' has the wrong address of the variable. The code generated by the compiler is correct.

Work around:
Remove the DATA statement, it is not needed on a Sun when the array is being initialized to all 0's.

Reference Number: 1002658
Synopsis: character strings assigned to dimensioned variables causes f77 error
Release: 3.2alpha

Description:
The following code has conversion errors, but should cause the 3.2PILOT f77 compiler only to generate an error message. However, the compiler dies giving the following error message:

```
 Compiler error
```

The code:

```
subroutine plabak
    dimension  wprt(3)
    wprt(1)  = ' del'
    wprt(2)  = 'p/pm'
    wprt(3)  = 'ax  '
    thetr    = thetar*180./pi
return
end
```

The error messages are:

```
f77 -O -c smalbug.f
smalbug.f:
      plabak:
Error on line 3 of smalbug.f: impossible conversion
Compiler error line 3 of smalbug.f: Impossible tag error
      in routine map_fortnode
```

Reference Number: 1003404
Synopsis: f77pass1 seems to infinitely loop on program
Release: 3.2

Description:
This problem occurs in programs which have a very large number of EQUIVALENCE statements which overlap the same variable. The algorithm used to compute the overlap uses a lot of memory which it does not free.

Work around:
Breaking up the subroutines into separate files will help if more than 1 subroutine has equivalence statements. Changing the EQUIVALENCE statements so that they do not overlap will also help. For example:

```
COMMON /CBLOCK/ ARR(10000)
EQUIVALENCE (X1,ARR(1)), (X2,ARR(100)), (Y1,ARR(5001)), (Y2,ARR(7000))
```

can be changed to:

```
COMMON /CBLOCK/ ARR1(5000), ARR2(5000)
EQUIVALENCE (X1,ARR1(1)), (X2,ARR1(100)), (Y1,ARR2(1)), (Y2,ARR2(2000))
```

Note: This work around will not be helpful in all cases.

Reference Number: 1003554
Synopsis: SU format control broken for integer*2
Release: 3.2, 3.0

Description:

The following FORTRAN program shows incorrect results when it uses the 'SU' format descriptor in a format statement to perform an unsigned assignment of a variable declared as 'integer*2' to an 'integer*4' value:

```
      integer *2 x
      x=-1

      write ( *,10) x
10     format ("The value is", I10)

      write ( *,20) x
20     format ( "The value is", SU, I10)

      write ( *,30) and ( x, 65535 )
30     format ( "The value is", SU, I10)
      end
```

The results for release 3.0 and release 3.2 are:

```
zeus>> a.out
The value is         -1
The value is4294967295
The value is      65535
```

Work around:

The output of the third 'write' statement in the FORTRAN program above shows that the workaround is to 'and' the variable or expression with 65535 to the i/o statement.

Reference Number: 1003626
Synopsis: Missing endif causes f77 to give bad label to assembler
Release: 3.2, 3.2alpha, 3.0

Description:
Omitting an 'endif' statement in a series of nested 'IF' statements causes the assembler and linker to give misleading error messages such as the following:

```
as: warning (/tmp/f77pass1.7117.d.2.s:701): Undefined L-symbol
Undefined:
L47
```

Reference Number: 1003813
Synopsis: Statement function causes assembler error
Release: 3.2

Description:
The statement function on the third line of the following FORTRAN program causes an error in the assembler code that is output from the program's compilation:

```
subroutine init
real a
junk(zz) = sin(26.0 - zz)
a = junk(0.0)
return
end
```

The following shows the compilation and the resulting console messages:

```
zues>> f77 t.f
t.f:
     init:
as: error (/tmp/f77pass1.15992.i.1.s:32): Invalid character
as: error (/tmp/f77pass1.15992.i.1.s:32): Invalid operand
```

Work around:
Use a subroutine rather than a statement function in programs.

Reference Number: 1003940
Synopsis: backslash in data statement causes f77pass1 to core dump
Release: 3.2

Description:
The FORTRAN compiler incorrectly treats a single backslash in a Hollerith specifier of a 'DATA' statement as an escape

character causing programs to core dump. The following program includes a backslash such as this in the 'data' statement, and core dumps with a segmentation fault when run:

```
subroutine b
     integer*4 jbar
     data    jbar/1h\/
     end
```

Work around:

Use '1H\\' instead of '1h\' or use a string constant instead of a Hollerith field.

Reference Number: 1003942
Synopsis: error message gives no clue as to program's actual error
Release: 3.2

Description:

When the FORTRAN program below is compiled, 'f77pass1' gives no error message to indicate that Sun's FORTRAN compiler does not allow an error label to point to a format statement:

```
    program a
    i=1
    write (6,22,err=10) i
22  format(i6)
10  format(i6)
    end
```

The assembler and linker catch the error, but give the following misleading warnings:

```
a.f:
 MAIN a:
as: warning (/tmp/f77pass1.892.d.2.s:80): Undefined L-symbol
Undefined:
L16
```

Reference Number: 1003979
Synopsis: extra spaces generated when writing to internal files
Release: 3.2

Description:

The problem with the following program is that on list directed output two extra blanks are inserted preceeding a character value:

```
common / x / label(80)
character*80 lab
character*1 label(80)
equivalence (lab,label)
real time
time = 7.8
write(lab,131) time
131    format(f7.3)
print *,label,'helen'
end
```

Here's the output which shows the problem:

```
      7 . 8 0 0
```

Work around:

Equivalence a character string to the array which is as long as the array and write the string instead of the array.

Reference Number: 1004008
Synopsis: f77 -i2 breaks logicals
Release: 3.2

Description:

A logical test that works correctly under FORTRAN compilation without options, fails when the '-i2' option is specified.

Work around:

Omit the '-i2' option if you are testing logicals.

Reference Number: 1004009
Synopsis: f77 -i2 breaks inquire( ) existence test
Release: 3.2

Description:

When the following FORTRAN program 'inq.f' is compiled with the '-i2' option, the inquire( ) does not work properly.

```
logical l
inquire(file='inq.f', exist=l)
if(l) then
      print*,'exists'
else
      print*,'nonexistent'
endif
stop
end
```

Run after each compile. Notice that the first time the file exists and the second time it does not.

Work around:

Change the program to pass the first element of a logical array and then test the second element. The example program may be changed as follows:

```
logical l(2)
inquire(file='inq.f', exist=l(1))
if(l(2)) then
     print*,'exists'
else
     print*,'nonexistent'
endif
stop
end
```

Reference Number: 1004051
Synopsis: f77 -i2 breaks integer parameters in comparisons
Release: 3.2

Description:

An integer parameter set to 2 has a value of 2 when printed out, but acts as if it has a value of 1 in comparisons. These comparisons fail with the -i2 flag and work without it.

Work around:

Omit the '-i2' option in this case.

Reference Number: 1004073
Synopsis: f77 rewind breaks write on newly-created file
Release: 3.2

Description:

A bug shows up in a FORTRAN program that opens a file and rewinds it. The program gets the corresponding UNIX file descriptor via 'getfd(3F)' and calls a C routine to do a 'write(2)'. If the file is not created prior to running the program and a 'rewind' is done, the 'write' does not work and an empty file is created. Without 'rewind' or if the file previously exists, the code works correctly.

Work around:

Do not use rewind on a non-existent file. Open the file, close it, and then re-open it.

Reference Number: 1004168
Synopsis: f77 doesn't allow opening all 30 file descriptors
Release: 3.4beta, 3.2

Description:
When attempting to use all thirty file descriptors available in a Fortran program, a runtime error occurs and the program aborts.

Reference Number: 1004169
Synopsis: Unix read of f77-opened file fails with "bad file number"
Release: 3.4beta, 3.2

Description:
When a new file is opened in Fortran and the file descriptor is passed to a C routine, the C routine attempts to write to the file, rewind with 'lseek', then read what it had just written. This fails, producing the following error message:

```
Read: Bad file number
errno=9
```

Work around:
Open the unit, close it, and then re-open it.

Reference Number: 1004177
Synopsis: Cannot have variable named "units" in mixed C and FORTRAN
Release: 3.4beta, 3.2

Description:
A fatal runtime error occurs when a program with mixed C and FORTRAN statements uses 'units' as a global variable name.
Work around:
Do not name a global variable 'units' in the C portion of a mixed C and FORTRAN program.

Reference Number: 1004206
Synopsis: 'dbxtool' won't display source for f77 .F files
Release: 3.4beta, 3.2

Description:
If the user compiles the following program with "f77 -g foo.F" (invoking the C preprocessor), and executes "dbxtool a.out", 'dbxtool' comes up with the messages "No Source Displayed" and

"can't exec <program.f>" in the command window:

```
program foo
i = 5
j = 6
end
```

Work around:

A workaround for the bug is to use following the command:

```
f77 -g -Qoption f77pass1 -Efoo.F  foo.F
```

rather than:

```
f77 -g foo.F
```

Reference Number: 1004348
Synopsis: f77pass1 core dumps if parameter used in parameter statement
Release: 3.2

Description:

When a subroutine parameter is used in a FORTRAN 'PARAMETER' statement, 'f77pass1' core dumps rather than reporting an error and continuing compilation.

Reference Number: 1004485
Synopsis: fortran x**y not correct for integral valued real y
Release: 3.2

Description:

The compiler attempts to optimize the expression on line 5 of the FORTRAN program, 't.f', below by squaring the left side of the '**' operator. When the left is an integer, and the right is a real, the integer should be converted to a real; however, this is not done.

```
C... FORTRAN program 't.f'
      program tstex1
      integer answer
      integer arg
      arg = 3
      answer = sqrt(arg**2.0)
      end
```

The following shows the compilation of 't.f' and the resulting error messages:

```
zeus>> f77 t.f
t.f:
 MAIN tstex1:
Error on line 5 of t.f: bad argument type to intrinsic sqrt
Compiler error line 5 of t.f: Impossible tag error in routine
map_fortnode
```

Reference Number: 1004671
Synopsis: f77 -O generates incorrect values within a loop.
Release: 3.2

Description:

The following FORTRAN77 program seqment shows a compiler optimization bug, with the generated code failing to save the "m-1" value into m1 on all but the first loop:

```
      parameter (ndim=5)
      real xx(ndim)
      data nn/ndim/
      data xx/1.0,2.0,3.0,4.0,5.0/
      call aaa(xx,nn)
      stop
      end
      subroutine aaa(x,n)
      real x(*)
      integer n
      y=1.0
      do 1 m=n,1,-1
      m1=m-1
      write(*,9001) m1
      if(y .le. 0.0) goto 1
      z = x(m)
      write(*,9002) z
1      continue
      return
9001  format('m-1= ',i3)
9002  format('z = ',f6.2)
      end
```

Reference Number: 1005335
Synopsis: no error reported for characters after closing ')' in format statement
Release: 3.2

Description:
The FORTRAN77 compiler does not report a syntax error for characters following the closing ')' in a 'format' statements.

Library

Reference Number: 1000077
Synopsis: "getwd()" library routine
Release: 3.0

Description:
The getwd() routine in the standard C library uses "stat()" rather than "lstat()" to search for the component segment names in the path to the current working directory. Most of the time, due to the normal order of file creation in a directory, getwd() finds the real directory entry first. But on occasion it will find a symbolic link to the directory, and by using "stat()" rather than "lstat()" will decide to report the symbolic link as the "correct" pathname component. In addition to causing even more randomness in the result of a getwd() call, this usage interacts badly with symbolic links to NFS mount points when a particular NFS server is not responding.

Reference Number: 1004297
Synopsis: putw() returns ferror instead of word putted.
Release: 3.2

Description:
putw(word,stream) returns an integer value which is returned by ferror(). putw() should return the word that was written out to the stream. For example:

```
main() {
        int x;
        FILE*   file;
        int c='47';

        if (file1 = fopen("myfile",RW) == NULL) exit;
        x = putw(c, file1)
}
```

The returned value for this successful putw( ) call was '0', when the user expected '47' based on the Kernigan and Ritchie standard.

This is a documentation bug because the value of 'ferror' would be the correct return value for 4.2BSD.

Reference Number: 1004577
Synopsis: Printf padding strings with leading zero's is broken.
Release: 3.2

Description:

In the 'UNIX Interface Manual', 19 September 1986, the documentation for printf(3S) states the following:

If the field width for an s conversion is preceded by a 0, the string is right adjusted with zero-padding on the left.

However, this feature is not supported by Release 3.2.

Reference Number: 1004834
Synopsis: Alarm going off during function call trashes results.
Release: 3.2

Description:

The C program below eventually fails when it is run with the alarm( ) call. During the call to 'sprintf( )', 'SIGALRM' is received and the program control is transferred to the signal handler. When control returns from the signal handler, the stack has been changed; hence, the program gives incorrect results.

The following first gives the C program, and second gives the commands and the C shell script used to run it:

```
#include <stdio.h>
#include <signal.h>

extern int sigfunc( );

main(argc,argv,envp)
        register int argc;
        register char **argv;
        register char **envp;
```

```
{
        register int i;
        char s1[50], s2[50];

        signal(SIGALRM,sigfunc);
        alarm(1);

        for(i = 0 ; i < 1000000 ; i++)
        {
                sprintf(s1,"%d",10);
                sprintf(s2,"%d",10);
                if (strcmp(s1,s2))              {
                        printf("%d\n",i);
                        puts(s1);
                        puts(s2);
                        exit(1);
                }
        }
        exit(0);
}

sigfunc()
{
        alarm(1);
}

======================================================

#1 cc -o SO81421 SO81421.c
#2 SO81421.csh
64425
10
14
ERROR : sprintf()
#3

======================================================
#!/bin/csh
while (1)
        SO81421
        if ($status) then
                echo "ERROR : sprintf()"
                exit 1
        endif
end

======================================================
```

Linker

Reference Number: 1005063
Synopsis: ld -A produces symbol tables that contain incorrect values
Release: 3.2

Description:

The symbol table that is output from the command 'ld -A' contains incorrect values. However, these incorrect symbols are not needed during compilation, execution, or debugging of the program.

Lint

Reference Number: 1002091
Synopsis: lint misses structure mismatches.
Release: 3.0

Description:

Running 'lint' on a program which passes a structure to a routine that expects a different structure of the same size does not generate a diagnostic. This contradicts the Type Checking section of 'C Language Tools' chapter of the Release 3.0 'Programming Utilities Manual' that states that 'all actual arguments must agree in type with their declared counterparts'.

Reference Number: 1002841
Synopsis: lint "-n" flag can't be bundled with other flags
Release: 3.2

Description:

"lint" has a "-n" flag that tells it not to check functions against the definitions in the "lint" library for the standard C library. In the old "lint" shell script, the "-n" flag could be bundled with other flags, so that "lint -hbxn" would set the "-n" flag as well as the "-h", "-b", and "-x" flags. This does not work with the new "lint" command, implemented as a symbolic link to "/lib/compile".

Work around:

Keep the "-n" option separate; in the case given, use "lint -hbx -n test.c" instead of "lint -hbxn test.c".

Reference Number: 1004471
Synopsis: Lint library def. for "openlog"
Release: 3.2

Description:

The 'lint' library description for the routine 'openlog' indicates that it returns no value. The manual entry for 'openlog(3)' indicates that it returns '0' on success; however, the code actually returns '0' on success and '-1' on failure.

This causes lint to report erroneous errors in programs that check for the return value.

Optimizer

Reference Number: 1003153
Synopsis: Fortran code generator bug
Release: 3.2

Description:

The following Fortran program works correctly when compiled without the optimizer option (-O) and incorrectly when compiled with the optimizer.

```
      integer xsen, ysen
      real c(2,1), cm(2,1)
      c(1,1) = 1.0
      c(2,1) = 2.0
      xsen = 1
      ysen = 2
      j = 1
      cm(xsen,j) = c(xsen,j) + c(ysen,j)
      cm(ysen,j) = c(ysen,j) - c(xsen,j)
      write (*,*) cm(1,1), cm(2,1)
      end
condition          behavior
-------------------------------------------------------------
compiled without "-O"   cm(1,1) set to 3; cm(2,1) set to 1
  under 3.0, 3.1 (not
  tested), or 3.2
compiled under 3.0 or   cm(1,1) set to 0; cm(2,1) set to 1
  3.1 (3.0 verified,
  3.1 not) with "-O"
Compiled under 3.2      cm(1,1) set to 0; cm(2,1) set to 1
  with "-O", run on
  Sun-3/50
Compiled under 3.2      crashes with segmentation violation
        with"-O", run on Sun-3/260
```

Work around:

Note that the above code will compile correctly with the -O option if the order of the operands for the addition is changed. In other words, change the following line:

```
cm(xsen,j) = c(xsen,j) + c(ysen,j)
```

to this:

```
cm(xsen,j) = c(ysen,j) + c(xsen,j)
```

Reference Number: 1003602
Synopsis: fortran optimizer causing IOT trap errors
Release: 3.2, 3.0

Description:

Occasionally, the Fortran optimizer fails and gives IOT trap errors due to complicated expressions in the Fortran code.

Work around:

Splitting the complicated expressions into parts with assignments to local variables works. For example,

```
    sx(n) = (rs(is1+n1-1)+2.5*sx(n1)-1.5*sx(n1-1))*0.5
```

could be rewritten as

```
  kludge = is1+n1-1
  sx(n) = (rs(kludge)+2.5*sx(n1)-1.5*sx(n1-1))*0.5
```

A safer workaround is to make the broken subroutine into a separate compilation unit, and compile separately without -O.

Reference Number: 1004841
Synopsis: fortran optimizer bug
Release: 3.2

Description:

When the following FORTRAN program is compiled with -O, it generates the wrong results:

```
integer ndtfil, ndttab
integer ntabfl(15), tabptr(15)

ndtfil = 0
ndttab = 5
tabptr(1) = 0

ndtfil = ndtfil + 1
```

```
        ntabfl(ndtfil) = ndttab - tabptr(ndtfil)
        tabptr(ndtfil) = tabptr(ndtfil) + 1

        print *, 'ndtfil = ', ndtfil, 'ndttab = ', ndttab
        print *, 'ntabfl(ndtfil) = ', ntabfl(ndtfil)
        print *, 'tabptr(ndtfil) = ', tabptr(ndtfil)

        stop
        end
```

Running the program gives the following results:

```
ndtfil =   1 ndttab =   5
ntabfl(ndtfil) =   0
tabptr(ndtfil) =   1
```

ntabfl(ndtfil) should be 5.

Reference Number: 1004930
Synopsis: optimizer using expression 10-I as I-10
Release: 3.4beta, 3.2

Description:
Infrequently, the optimizer switches the order of operands in expressions passed as arguments to procedures in the file of assembler instructions that are output from compilation of a FORTRAN program.

Reference Number: 1004995
Synopsis: Optimizer uses a4 instead of d5
Release: 3.4beta, 3.2

Description:
Infrequently, the optimizer substitutes the register 'a4' rather than the register 'd5' in the file of assembler instructions that are output from compilation of a FORTRAN program. When this substitution occurs, the FORTRAN program behaves abnormally after optimization.

Utilities

Reference Number: 1002668
Synopsis: cpp doesn't handle 2 macros broken over lines
Release: 3.3, 3.2alpha, 3.2

Description:

The machine produced code shows a bug in the cpp processor. cpp cannot handle calls from a macro that are broken over two lines.

```
#define ASSIGN(x, y) x = y;

foo()
{
ASSIGN(a,
b->c);ASSIGN(j,
k->l);
}
```

it produces the following output:

```
% cc -E foo.c
# 1 "foo.c"


foo()
{
 a =  b->c;;# 7 "foo.c" <--- this is the bug
 j =  k->l;;
# 8 "foo.c"
}
```

Work around:

Don't break calls from macros over 2 lines. However, this is difficult when code is machine generated as in this case.

Diagnostics

Diagnostics

Reference Number: 1003714
Synopsis: problems using port B as console with diag switch on
Release: prom1.8and2.3

Description:
On the CPU for the 3/75, 3/140, 3/160, 3/180, the system can hang when the console is on port B and the diagnostic switch is on. Port A seems to function properly.
Work around:
1. "x" will work if preceded by a "ub" monitor command.
2. By typing a character in the 10 sec time period it will invoke the extended test system.

Reference Number: 1003926
Synopsis: bootproms do not check the contents of the eeprom for errors
Release: 2.0proms

Description:
On 3/260HM, if the EEPROM is set for low resolution (0x16 = 0x00) and monochrome display (0x1f = 0x00), after a k2 reset or a power cycle, the system comes up and displays bits skewed across the video monitor. The banner and boot messages are unreadable.

On 3/260C, if the EEPROM is set for high resolution (0x16 = 0x13) and color display (0x1f = 0x12), when a large text file (more than 1 screenful of text) is displayed using 'more', several lines more than a screenful are displayed, the first lines of the file being scrolled off the screen. If a large text file is edited using vi, lines disappear and reappear in the wrong places. Frequently, several blank lines will appear when there are none in the original file.
Work around:
Correct the contents of the EEPROM and power cycle.

Reference Number: 1004563
Synopsis: bootproms do not properly handle the checksum field of the eeprom.
Release: 2.3proms

Description:

The bootproms compute the ‘checksum’ field in the ‘eeprom’ incorrectly.

Work around:

Use ‘eeprom -c’ to fix the ‘checksum’ field.

Documentation

Documentation
FORTRAN

Reference Number: 1003106
Synopsis: f77(1) claims f77 leaves .o files by default
Release: 3.2alpha, 3.0

Description:
See Release 3.2 Commands Reference Manual (page 156), Section f77(1). The text reads as follows: "Filenames ending in .f are taken to be FORTRAN 77 source programs; they are compiled, and each object program is left in the file (in the current directory) whose name is that of the source with .o substituted for .f." Compiling actually does not produce a .o unless the -c option is used. The -c option is described later in the same document.

Reference Number: 1000500
Synopsis: The example when typed in as documented does not work
Release: 1.3

Description:
In the Programming Tools for the Sun Workstation, page 8, the following example appears:

```
1,$s/^[]*//
```

If the user is unfamiliar with ed, he or she will be unable to detect that a space is missing between the brackets.

Work around:
Change the documentation so that a space between the brackets is more evident.

Reference Number: 1003027
Synopsis: on(1C) bugs section should include ^Z (CTRL-Z) hangs windows
Release: 3.2

Description:
There is missing information in the bugs section of the UNIX command on(1C)--if on(1C) is running, entering a Control-Z hangs the window over NFS mounts.

SunCore

Reference Number: 1000510
Synopsis: details of using cgpixwindd from f77 suncore missing.
Release: 2.0

Description:

SunCore manual does not adequately explain the FORTRAN interface to the vwsurf struct. In particular, when using cgpixwindd, the cmapsize element of the vwsurf struct must be set if it needs to be other than the default of 2.
The resulting application runs fine on raw color surfaces, but appears much like a black and white application when run in a window.
Additionally, the documentation does not show the f77 equivalents of the C vwsurf struct.

SunView

Reference Number: 1004829
Synopsis: pw_get_region_rect(), pw_set_region_rect() descriptions incorrect
Release: 3.2

Description:

See the SunView Programmer's Guide, 15 October 1986. The return value of 'pw_get_region_rect()' on page 311 and 'pw_set_region_rect()' on page 313 is documented to be a 'Pixwin *'. These both return an 'int -', which indicates success or failure. For example, '-1' indicates a failure.

Work around:

The interfaces should be as follows:

```
int
pw_get_region_rect(pw, r)
        Pixwin *pw;
        Rect *r;

int
pw_set_region_rect(pw, r, use_same_pr)
        register Pixwin *pw;
        register Rect *r;
        int use_same_pr;
```

System Administration

Reference Number: 1000412
Synopsis: misleading info on restore in 2.0 Sys Admin manual, section 3.4.2
Release: 2.0

Description:

See Release 2.0 System Administration Manual, Chapter 3 - Disk and File Systems, Section 3.4.2 - Restoring An Entire File System, Step 4; and Section 3.4.3, Step 7.
This section does not indicate that 'restores' must be done with the same blocking factor as when 'dumped.' This is particularly important

when making multi-tape dumps to 1/4-inch tape. The System Administration Manual indicates use of 126-blocking factor on dumps, but does not mention specifying the same blocking factor during restores. When restoring a multi-tape, 1/4-inch dump tape created with a 126-blocking factor, the following error message appears at the end of a tape:

```
partial block read: <num> should be <num>
```

The contents of this partial-block read is not copied onto disk.

Work around:

Add the appropriate info to the mentioned section, e.g.,

4) Change to the /mnt directory and restore the level zero tape.

For a 1/2-inch tape drive:

```
# cd /mnt
# /etc/restore rvf /dev/rmt0
```

And for 1/4-inch tape drives (substitute for rdrive one of the following - rar0 for archive tape controller; rst0 for SCSI tape controller):

```
# cd /mnt
# /etc/restore rvbf 126 /dev/rdrive
```

Continue to restore incremental tapes...

Reference Number: 1000420
Synopsis: ACUHAYES omitted from supported device types for UUCP documentat
Release: 2.0

Description:

ACUHAYES is omitted from supported-device types for uucp in the manual System Administration for the Sun Workstation, "Tutorials" index tab, "uucp Implementation Description" chapter, Section 9.10, Device Types (page 18).
The table shown should have a line with:

ACUHAYES	Hayes Smartmodem 1200

There should be an additional note indicating that Sun supports the Ven-Tel 1200 PLUS (EC1200-32) and EC1200-31 (when set for 'AT' command recognition) with the ACUHAYES device type. Note that some are AT-compatible and some are not.

Reference Number: 1000656
Synopsis: 'Setting Up A Gateway Machine' omits adding new /etc/host entry.
Release: 3.0, 2.0

Description:

See Release 3.0 System Administration manual (pages 106-107), Communication chapter. This section discusses setting up a gateway machine, but has omitted one important step: after adding the new hosts entry to /etc/hosts on the Yellow Pages master server and running 'make hosts', the gateway machine must also have the new host's entry included in its own /etc/hosts file. The reason for this is that the /etc/ifconfig command(s) are run from /etc/rc.boot before YP starts and requires this information.

Reference Number: 1003286
Synopsis: there should be a comment in uucp.h for UUDIR define
Release: all

Description:

See the System Administration manual (page 316). A reference is made to UUDIR being a define in uucp.h. UUDIR is actually a CFLAG.

Work around:

For additional information on the format CFLAGS and UUDIR see src/sun/usr.bin/uucp/Makefile.

Reference Number: 1003362
Synopsis: errors in sample printcap section of adding hardware chapter
Release: 3.0

Description:

See Release 3.0 System Administration manual (pages 160-161). In the sample printcap entry on page 160, the presence of both the fs/fc values and the xs are counter-productive. xs#040 sets LLITOUT, which disables all output processing. This undoes what is attempted by the fs#06020 and fc#0300, which tries to set XTABS, CRMOD, and space parity, but is overridden by the LLITOUT feature. On page 161, the 'Clear flag bits' printcap capability should be named 'fc' and not 'fs'.

Work around:

Remove the 'xs#040' field from the sample printcap entry.

Reference Number: 1003444
Synopsis: 19 field L.sys line limit not documented
Release: 3.0

Description:
L.sys lines have a 19-field limit.

Reference Number: 1003555
Synopsis: updates for 'Adding A Modem...' to 3.x kernel configuration
Release: 3.0

Description:
See the System Administration manual (pages 151-152), Revision B of 17 February 1986, Chapter 5 - Adding Hardware To Your System, Section 5.4 - Adding A Modem To Your System.

The references to kernel configuration lines for the zs0 device are out of date. Specifically, the manual shows the following line:

```
device          zs0 at mb0 csr 0xeec800 flags 0x3 priority 2
```

Work around:
Use the current line from a Sun-3 3.x configuration file:

```
device          zs0 at obio ? csr 0x20000 flags 3 priority 3
```

Reference Number: 1003917
Synopsis: Sys. Admin. manual doesn't mention 60 Mb tape
Release: 3.2

Description:
See the System Administration manual, "Backing Up File Systems with Dump," Section 3.3. There is no mention of 60Mb quarter-inch tapes, and the computation of dump's "tape length" parameter for quarter-inch tapes is not clear.

User Manual

Reference Number: 1002963
Synopsis: getpwent(3) does not discuss yellow pages
Release: 3.0

Description:
See the UNIX Interface Reference Manual, 15 April 1986, 'getpwent(3)' manual entry, page 198. The manual entry does not mention that 'getpwent(3)' looks up entries in the yellow pages database, and interprets the '+' automatically.

Reference Number: 1003134
Synopsis: send(2): man page should state what sockets SOF_OOB work on
Release: 3.2

Description:
The man page for send(2) does not state what socket types support OOB data. Attempting to send and receive OOB data on an AF_UNIX SOCK_STREAM causes the system to panic with an mfree error.
Work around:
Do not use OOB data on an AF_UNIX SOCK_STREAM.

Reference Number: 1003460
Synopsis: telnetd(8C) incorrectly reports 16-pseudo-tty limit.
Release: 3.2

Description:
The 3.2 telnetd(8C) man page states the following under the BUGS heading:
"telnetd can only support 16 pseudo terminals"
This restriction applies only to the pre-3.2 releases. The 3.2 version of in.telnetd has been modified, making it capable of handling up to 64 pseudo-ttys.

Reference Number: 1003721
Synopsis: Manual page error for pwck, grpck
Release: 3.2

Description:
See the man pages for pwck(8) and grpck(8). The text states these utilities are located in /etc, but these utilities are actually located in /usr/etc.

Reference Number: 1003805
Synopsis: passwd(5) nor adduser(8) discuss legal uid and gid values
Release: 3.2

Description:
See the UNIX Interface Reference Manual, 19 September 1986, descriptions for 'getgid(2)' on page 47, 'getuid(2)' on page 64, and 'passwd(5)' on page 556; and the UNIX Commands Reference Manual, 19 September 1986, description for 'adduser(8)' on page 558. The documentation indicates that user ids and group ids are both specified as non-negative 16-bit integers, but it does not indicate what the legal range of values are.

Work around:

The values range from 0 through 32767, with the exception of the special use '-2 nobody' user id.

Reference Number: 1004052
Synopsis: 8 significant characters in password mostly undocumented
Release: 3.2

Description:

The number of significant characters in a password is 8. This is not documented in the expected places, but can be found in the manual page for getpass(2).

Reference Number: 1004242
Synopsis: kadb cannot be used on diskless machines
Release: 3.2

Description:

See man page kadb(8S). kadb cannot be used on diskless machines to get to the boot prompt "kadb>".

Work around:

Enter the following:
```
> b kadb -d
```
After the messages appear, the following is returned:
```
kadb:
```
Enter the following, then press <CR>:
```
le(,,1)vmunix
```

Reference Number: 1004455
Synopsis: route(8c) does not match usage line
Release: 3.2

Description:

See the Commands Reference Manual, 19 September 1986, 'route(8c)' description, page 675. The 'route' usage line includes the '-h' and '-n' options, as follows:

```
usage: route [ -f ] [ -h ] [ -n ] [ cmd args ]
```

These options are not included in the 'route' man page, which appears as follows:

```
/usr/etc/route [ -f ] [ command args ]
```

Graphics

Graphics
cgi

Reference Number: 1002597
Synopsis: cgi: cgipw does not respect pixwin regions
Release: 3.0

Description:
CGIPW does not respect pixwin regions.
The size (rect) of the pixwin is determined by calling win_getsize( ) on the pixwin window fd, rather than using pw_getregionrect( ), or using any size info actually in the pixwin struct itself.

The SunView canvas window deals with region pixwins. The pixwin handle which is made available is usually a pixwin region of the full pixwin of the canvas window. Since scrollbars are implemented as pixwin regions, it is mandatory that these region areas be preserved. This mismatch between CGIPW and SunView canvas use of pixwin regions prevents the integration of CGIPW on a SunView canvas which has scrollbars.

Reference Number: 1002598
Synopsis: cgipw: retained pixrect must be same size as screen pixrect
Release: 3.2alpha, 3.0

Description:
CGIPW requires that the retained pixrect be the same size as the screen pixrect of the pixwin of the cgi view surface. However, a SunView canvas may have a backing (retained) pixrect
larger than the visible screen window area. This allows scrolling around a large image with a smaller window.

This mismatch prevents the use of CGIPW on a SunView canvas which has a retained area larger than the visible window (and may have scrollbars).

Reference Number: 1002679
Synopsis: Re: Color map is not reloaded when entered from other screen
Release: 3.0

Description:
Color map is not reloaded when entered from other screen.
Work around:
Popping up a menu (anywhere) and bringing it down without invoking anything will reload the right colormap.

Reference Number: 1002999
Synopsis: cgi: arcs are sometimes drawn rotated about endpoints
Release: 3.0

Description:
Arcs are sometimes drawn rotated about the arc endpoints. Thus, an arc having endpoints with the same y value that should be drawn so that it curves up, is drawn so that is curves down.

If the amount of physical screen space is large enough, the program will seg fault in pr_curve(). This usually works if the window is fullscreen, or on the raw device.

Reference Number: 1003386
Synopsis: Ccgiwin descriptor causes CGIPW primitives grief
Release: 3.2

Description:
The efficiency and correctness of several CGIPW functions are compromised by the structure of the Ccgiwin handle that clients use to communicate with output primitives and attribute setting functions. It currently contains a pixwin handle and an attribute pointer, but not a view surface pointer. The view surface pointer can be found inefficiently by an internal CGI function that linearly searches the view surface table, but some functions that should call this function do not, and the ones that do call it suffer a performance penalty by having to do so.

Reference Number: 1003455
Synopsis: cgipw_set_vdc_extent fails if using more than one view surface
Release: 3.2

Description:
Cgipw_set_vdc_extent (a new SunCGI extension in 3.2) will only work correctly if just one view surface is in use, because it doesn't set the global _cgi_output_att pointer that determines which scaling parameters are used by the _cgi_devscale function. Since _cgi_devscale is used by _cgi_windowset, which is called by cgipw_set_vdc_extent to set scaling, it will use the current _cgi_output_att, which will have been left set correctly by open_cgi_pw when the single view surface in use was opened.
Work around:
Call some other cgipw_ function that takes a descriptor as an argument and has no undesirable side effects (cgipw_inquire_text_attributes is a good choice) with the descriptor of the desired view surface, then call cgipw_set_vdc_extent with that same descriptor. The global value _cgi_output_att will have been left set by the cgipw_inquire_text_attributes call, and will be used correctly.

gp

Reference Number: 1004863
Synopsis: GP1_PR_PGON_TEX problem.
Release: 3.2

Description:
When using GP1_PR_PGON_TEX to draw polygons, the polygons get drawn incorrectly on the GP when part of the polygon intersects the left edge of the screen (NOT the window). If a window goes from 100 to 400 in X, any polygons that have coordinates of less than -100 (the edge of the physical screen) get clipped incorrectly (they are pushed to the right). The scan lines used to fill the polygon are the correct length, just moved to the right. Polygons that fit on the physical screen are clipped correctly to the window boundary; only those that if not clipped would extend off the physical screen are affected.
Work around:
Clip polygons to the screen in applications, and supply a negative 'x' component in the 'sx' field

Reference Number: 1005359
Synopsis: pw_line and pw_polyline bug.
Release: 3.4, 3.2

Description:
When a user starts 'Suntools' by default on a system with 'gp' and uses 'pw_line' or 'pw_polyline' to draw a vector from left to right whose starting point has a negative 'x' coordinate, nothing happens. However, if the user starts 'Suntools' with the option '-d /dev/cgtwo0', these two functions work fine. To be affected, the line must be textured or have a width greater than 1.
Work around:
Pre-scan the polyline vertices for negative 'x' values; decompose problem polylines into vectors and pass the positive 'x' coordinate first.

pixrect

Reference Number: 1005215
Synopsis: pw_getattributes() returns differently on cgfour
Release: 3.2

Description:
The 'pw_getattributes()' call can return a different value for planes when running on the 'cgfour' than it does for the 'cgtwo'. In particular, on 'cgtwo' it will return 0xf for a colormap segment of size 16, but on cgfour it gives 0x400000f.
Work around:
Mask out any bits above 255 (ff).

SunCORE

Reference Number: 1000905
Synopsis: SunCORE: text does not transform correctly
Release: 3.2, 2.0

Description:
In SunCORE text written to a segment which is to be transformed does not properly clip to the window boundaries, even with both window and output clipping enabled.

Reference Number: 1000917
Synopsis: segment containing text is not clipped properly when scaled
Release: 3.2alpha, 3.0

Description:
In SunCORE when scaling a segment containing text with the routine set_segment_image_transformation_2(), the graphics (lines) seem to clip properly, but the text appears to wrap at some point.

Reference Number: 1003703
Synopsis: suncore: only allows 20 fds
Release: 3.2

Description:
SunCORE does not read input if the file descriptor of the vwsurf.windowfd field is >= 20.
Work around:
Use a fd < 20.

Reference Number: 1004067
Synopsis: suncore: GP ucode breaks in 3.2, worked in 3.0
Release: 3.2

Description:
Using the 3.2 Graphics Processor microcode with SunCORE frequently fails. Window boundaries are not preserved and bands of color are drawn both inside and outside the window.
Work around:
Turn output clipping off in the program. When using the GP the clipping is not much overhead anyway.

Reference Number: 1004150
Synopsis: inqcurrpos2 fails for SunCore called from Pascal
Release: 3.2

Description:
In SunCORE called from Pascal, inqcurrpos2 fails and causes the application to core dump with a memory fault. However, calls to inqcurrpos3,passing in a dummy variable as the third parameter.
Work around:
Always use inqcurrpos3.

Reference Number: 1004254
Synopsis: await_any_button_get_locator_2( ) returns incorrect location.
Release: 3.2, 3.0

Description:
await_any_button_get_locator_2( ) sometimes returns an incorrect location. This usually occurs directly after the mouse is moved into the viewport and a button is pressed.
Work around:

Move the mouse around in the viewport for awhile before pressing any buttons.

Kernel

Kernel
Driver

Reference Number: 1001028
Synopsis: system runs out of I/O buffers if kernel sends too many error msgs
Release: 2.0

Description:
When running page mode on a console window, if the kernel sends many error messages to the console, the system runs out of I/O buffers and hangs.

Reference Number: 1004153
Synopsis: 3.3 is causing bus error panics when accessing via nfs
Release: 3.3

Description:
Sun OS 3.3 produces a bus error when accessing a file/program via NFS. This problem occurs when starting the Sunlink Internetwork Router on a cpu port. When starting up the remote end of the link then the local end of the link, a panic occurs on the local system.

Reference Number: 1004247
Synopsis: cmdtool: does not always echo shell prompt
Release: 3.4beta

Description:
The shell prompts in a cmdtool are not always echoed. This happens primarily when hitting the RETURN key quickly, or holding it down. The prompts that are echoed do not match the number of times RETURN is hit.

Reference Number: 1004503
Synopsis: Serial port device drivers do not check tty line discipline in use
Release: 3.2

Description:
The serial port device drivers still make direct calls to 'tty' line discipline routines for the old/new discipline without first checking to ensure that the line in question is actually

running one of these disciplines. If the port is using a line discipline where the raw or canonical input queues may not be clists, this will cause a kernel panic.

Reference Number: 1005241
Synopsis: Booting from 1/2" tape (tm/cdc) on a 3/260 hangs system.
Release: 3.2

Description:

When attempting to boot Release 3.2 on a Sun 3/260 from a 1/2" tape, using a Tapemaster controller and CDC tape drive, the system hangs.
Work_around for BugsLIst:
The tape boot block must be first patched as follows:

```
>bmt()
Boot: mt(0,0,0)
Boot: <L1-A>
Abort at <some_address>
>l a18ae
000A18AE: 61FFFFFF? 4e714e71
000A18B2: FBD8504F? 4e71504f
000A18B6: 206EFFFC? q
>c
<cr>
Boot: mt(,,3)
```

Continue normal boot of 'diag' to format and label the disk. Next, boot the standalone copy to get 'miniroot' off the tape and onto the disk. Patch the standalone copy program as follows:

```
diag> q
Boot: mt(,,4)
Size: ...
Standalone Copy
From: <L1-A>
Abort at <some_address>
>l 563a
0000563A: 61FFFFFF? 4e714e71
0000563E: FBD8504F? 4e71504f
00005642: 206EFFFC? q
>c
mt(,,5)
To: <disk>(,,1)
```

General

Reference Number: 1001124
Synopsis: After remote /etc/shutdown, console is still in "raw" mode.
Release: 3.0, 2.0

Description:
After remote /etc/shutdown, console is still in "raw" mode.
Work around:
The console terminal is still working, but it is in raw mode so you can type:

```
# stty echo -raw<lf>
#
```

Note that you won't see this displayed, but you will get another prompt.

Reference Number: 1001154
Synopsis: SIGIO doesn't work well with pipes
Release: 3.2beta

Description:
In a two-process program using SIGIO that is connected by a pipe (where the first process sets up a SIGIO handler that will read stdin when a string appears on it, and the second writes a string on stdout every five seconds), when the second is piped into the first, the first process doesn't print any messages.

Reference Number: 1001149
Synopsis: Sun3/160C with 3.0FCS halts for no apparent reason
Release: 3.0

Description:
When running generic Sun 3/160C, 4MB, SCSI disk and tape with several NFS file systems mounted, the system completely locks when the following is run:

```
tar tvbf 126 /dev/rst0
```

Attempts to interrupt tar fail, the tape doesn't move, and ping from a remote system succeeds, but rlogin times out.

Reference Number: 1001168
Synopsis: Degenerate filename problems in lookuppn():sys/vfs_lookup.c
Release: 3.2alpha

Description:
In Release 3.2Pilot, if an attempt is made to write to an existing directory with a degenerate filename (such as "." and "/"), the system won't allow the write, but it returns an incorrect error code -- EINVAL instead of EISDIR. In Release 3.0FCS, the system may return EPERM as the error code.

Reference Number: 1002485
Synopsis: Clocktool leaves pseudo-tty in an unusable state
Release: 3.2, 1.1

Description:
Clocktool leaves the pseudo-tty set to a state in which other terminals cannot set it to their environment type. The window programs that use pseudo-ttys need to either close them or lock them somehow so that they cannot be accessed by other programs using pseudo-ttys (inetd).
Work around:
An undocumented stty function in 3.2 can be used to reset rows and columns:
stty rows 0 cols 0

Reference Number: 1002579
Synopsis: kernel strlen() crashes system if passed null pointer
Release: 3.0

Description:
If the kernel version of strlen() is passed a null pointer, the system crashes.

Reference Number: 1002765
Synopsis: some ioctls in ioctl.h do not exist
Release: 3.2alpha, 1.1

Description:
Some of the 'ioctls' defined in '/usr/sys/h/ioctl.h' do not exist, and have not existed in Sun code since Release 1.1. Examples of some of these are as follows:

```
#define TIOCMODG   _IOR(t, 3, int)   /* get modem control state */
#define TIOCMODS   _IOW(t, 4, int)   /* set modem control state */
```

Some of the 'TIOCM_xxx' 'icotls' may also not exist, most notably 'TIOCM_CTS' and 'TIOCM_DSR'. The contents of the header files in the system is not documentation for what is supported in the system. The supported interface is described in the manual set, which is why these 'ioctls' are not described in the manuals.

Reference Number: 1002853
Synopsis: tar | tar, disk-to-disk on same xy450, crashes system
Release: 3.4beta, 3.2, 3.2alpha, 3.0

Description:
Performing a tar | (cd /fu/bar; tar) when the first directory is on one disk and /fu/bar is on another disk (and both disks on the same controller) crashes the system.
Work around:
If hardware and slots are available in the cardcage, put the disks on separate controllers.

Reference Number: 1002976
Synopsis: Socket application causes system to panic.
Release: 3.2, 3.0

Description:
Some Unix domain socket applications cause the system to panic.
Work around:
If possible, convert the affected applications to use Internet domain sockets in place of UNIX domain sockets. There should be no problem doing so unless the applications use the access rights passing feature of the UNIX domain to pass open file descriptors from one process to another.

Reference Number: 1003149
Synopsis: newly-installed system hangs with 'no space' message
Release: 3.2

Description:
The error is related to 3.2 installation. When using default setup parameters, a stream of the error message "no space" occurs after setup during boot of the system. The hardware configuration is:
3/160s with 8MB RAM
380MB drive on Xylogics 450
1/4" tape.
A similar hardware configuration can also cause this problem.

The disk is configured as follows:

A - 19.07	E - FREE
B - 16.62	F - 20.21
C - 131.17 (4 clients - root=8MB & swap=24)	G - FREE
D - 138.880	H - 51.21

Work around:

Change partitions, or reinstall, changing the disk configuration and/or client configuration. (Note that this may be a trial-and-error procedure.)

Reference Number: 1004165
Synopsis: setting raw mode under bk(4) line discipline panics system
Release: 3.2

Description:

Setting raw mode after having enabled the 'bk(4)' line discipline causes the system to panic with a bus error. A bug in the Berknet 'ioctl' routine incorrectly accepts the 'TIOCSETP ioctl', instead of rejecting them with an error code and diagnostic message.

Work around:

Set raw mode before setting the line discipline.

Reference Number: 1004323
Synopsis: Re-debugging prog with aborted open of pipe crashes system.
Release: 3.2

Description:

When re-debugging a program running under dbx (previously interrupted while opening a named pipe), the following occurs:

```
panic: bus error
```

Reference Number: 1004750
Synopsis: /usr/include/sys/uio.h doesn't prevent multiple include
Release: 3.2

Description:

Two portability problems occur between Sun's version of '/usr/include/sys/uio.h' and DEC's ULTRIX version. The Sun version does not define '_UIO_', thus making it unsafe against multiple includes, and the names for the values for 'uio_seg' are different, as follows:

Sun:

UIOSEG_USER
UIOSEG_KERNEL

DEC:

UIO_USERSPACE
UIO_SYSSPACE
UIO_USERISPACE

This difference is because the Sun version uses 4.2BSD terminology and the DEC version uses 4.3BSD terminology.

Work around:

Do not include 'sys/uio.h' more than once.

Reference Number: 1004768
Synopsis: Large maxusers causes 'sys pt too small' message when booting
Release: 3.3, 3.2

Description:

The system page table is too small for large values of 'maxusers'. Boot fails with the following message:

```
sys pt too small
```

The same error message is returned for large values of 'ntext'.

Work around:

Decrease the value of 'NPROC' defined in '/usr/sys/conf/param.c'. In the following, the value of NN is decreased:

```
#define NPROC (10 + NN * maxusers)
```

Reference Number: 1004782
Synopsis: 't_intrc' name conflicts in header files
Release: 3.2

Description:

There is a name conflict between a preprocessor definition and a kernel data structure definition. The conflict causes very bizarre compiler error messages, depending on the order in which the 'ioctl.h' and 'tty.h' header files are included in a C program.

The problem is that 't_intrc" is both an element in a structure definition in 'ioctl.h', and a '#define to a structure selection [different structures]' in 'tty.h'.

Work around:

Include header files in the following order:

ioctl.h
tty.h

Reference Number: 1004861
Synopsis: kernel allocates memory for credentials and doesn't free it
Release: 3.4, 3.2, 3.0, 2.0

Description:

The kernel sometimes neglects to decrement the reference count on a credential structure; thus, the structure is never de-allocated. This problem shows up on systems that have been up for a long time with heavy NFS usage and heavy 'setuid' program usage.

Work around:

Do not run 'setuid' programs repeatedly.

syscall

Reference Number: 1001256
Synopsis: putting /dev/zsmouse in async mode kills window system
Release: 3.0

Description:

The following code sequence causes a double panic: sleep when executed in a window:

```
fd = open("/dev/zsmouse",0);
fcntl(fd,F_SETOWN,getpid());
fcntl(fd,F_SETFL,FASYNC);
```

where /dev/zsmouse is character device major 12, minor 3. The same result occurs with /dev/mouse.

Reference Number: 1001271
Synopsis: ptrace interaction with interrupting slow system calls
Release: all

Description:

If a slow system call such as select or read from a terminal is interrupted while a program is under the control of a debugger, then the debugger is unable to call a function in the program being debugged.

When the slow system call is interrupted, the kernel does a setjmp to remember where it was. To call a function in the program being debugged, dbx writes a jsr instruction in the program's address space and with the ptrace option (SINGLE_STEP,) executes the jsr. Instead of executing the jsr, the kernel does a longjmp back into the interrupted system call and waits for it to complete.

Reference Number: 1002665
Synopsis: killing and restarting socket listener causes sender to hang
Release: 3.0

Description:
In 3.0 when UNIX domain sockets are used to communicate between processes, if the listening process is killed and then is restarted, the talking process hangs.
Work around:
Use Internet domain sockets.

Reference Number: 1003135
Synopsis: "panic: mfree" with AF_UNIX SOCK_STREAM OOB data
Release: 3.2

Description:
AF_UNIX SOCK_STREAM OOB data is not a supported feature of SunOS.
Work around:
Use OOB (out of band) data only with Internet domain sockets.

Reference Number: 1005116
Synopsis: writes to a pipe are not atomic
Release: 3.2

Description:
A kernel bug occurs in the following scenario:
A process writes to a pipe. The pipe becomes full, so the process sits in 'write(2)'. Then the process receives a signal. The signal handler also writes to the pipe. When a second process tries to read from the pipe, it gets garbage. The signal handler should not have been able to write to that pipe, because it was full.

The problem appears to be that writes to pipes are not necessarily atomic. Data from a write issued by one process may be interleaved with data written by another process.

Reference Number: 1005223
Synopsis: I/O from shared memory to raw disk produces inconsistent data on Sun-3/260, 3/280
Release: 3.2

Description:
Raw I/O to disk from shared memory produces inconsistent data on Sun-3/260 and Sun-3/280 systems, but works properly on Sun-3 systems without caches.
Work around:
Use block disk device, rather than raw disk device. An alternative is to move data from shared memory area to regular memory; then do an I/O from regular memory to raw disk device.

Network

Network
Library

Reference Number: 1002879
Synopsis: initgroups(3)/getgrent(3) causes malloc'd memory overwritten
Release: 3.2

Description:
If initgroups(3) is called two times, the second call results in the following error message from free, indicating that malloc'd buffers are overwritten:

```
free: bad block size (1919118906) at 0x27850
```

Examination of the core dump shows the offending call to free occurs in getgrent(3), which is called from initgroups.
The contents of the malloc'd memory are strings from the groups file.

This bug is reproduced when the test program is compiled with the debugging malloc module, /usr/lib/debug/malloc.o.

nfs

Reference Number: 1001294
Synopsis: nfs: root access across net does not check group access correctly
Release: 3.2alpha, 3.1, 3.0, 2.2

Description:
nfs does not correctly check group access for root of the nfs client if the group is NOT wheel.
If an nfs mounted file system has a directory with the following conditions, it is writable by root on the nfs client (i.e. writable by user nobody):
1. only has owner and group access permissions (i.e. mode 770)
2. owner is root
3. group is NOT wheel

If the group of this same directory is changed back to group wheel, root on the nfs client no longer has write permissions (as it should be).

Reference Number: 1001992
Synopsis: Can't use chgrp on an nfs mounted file system
Release: 2.0

Description:
It is not possible to use "chgrp" on a file in your home directory when logged onto a client machine which is nfs mounted. The error message received in this circumstance is:
filename not owner
If you rlogin to the server and repeat this command, it will work fine. This is due to chgrp being setuid root.

Reference Number: 1003161
Synopsis: nfs client caching has protection problems
Release: 3.2

Description:
Apparently, there is client side caching going on that latches [bad] permissions for files accessed over the nfs. If I have a file, eg., my mbox, that is mode 600, and I try to first read it as root, the access fails. However, if I then turn around and try and access it as myself, it fails as well. Likewise, if I access a file as myself, then root can see it too, until I try and remove it as root, then no one can see it.
Work around:
flush the cache to clear the condition
```
<hal> find . -name lambda -print
```

Reference Number: 1003673
Synopsis: rpc.mountd thrashes with large export list
Release: 3.2

Description:
If a server has a large list of machines to which a file system is allowed to be exported, the mount deamon thrashes while parsing /etc/exports. Apparantly, rpc.mountd is trying to get the network address of each machine in the exports list for each request. If there are a large number of requests (say after a power failure) it thrashes.
Work around:
Delete the machine export list. This removes protection, but allows the daemon to make progress.

Reference Number: 1004378
Synopsis: lockf(3) fails over NFS
Release: 3.3, 3.2

Description:
If a process on the NFS server has the file locked via 'lockf', and a process on the client mounting the file system (where the file resides) uses 'lockf' to test whether the file is locked, the test reports that the file is unlocked, even though it is still locked from the standpoint of the processors on the server.

Work around:
Use a non-blocking 'lockf' call to test. If the call succeeds, be sure to immediately unlock. This is functionally equivalent to test.

Reference Number: 1004496
Synopsis: mount(8) option intr doesn't allow keyboard interrupts
Release: 3.3, 3.2

Description:
The intr option to mount(8) doesn't allow keyboard interrupts to kill a process that is hung waiting for a response on a hard mounted filesystem.

Reference Number: 1004562
Synopsis: login and nfs get confused over group permissions
Release: 3.3, 3.2

Description:
When logging in to an NFS-mounted home directory, group permissions are incorrectly handled. 'login' runs as 'root' as it sets up the user environment, including 'cd' to the correct home directory. The NFS server maps 'root id (0)' to 'nobody (-2)'. This id is excluded from reaching the home directory; therefore, 'root' cannot get there.

Work around:
Make the mode 755, or have the NFS server allow root access.

Reference Number: 1005489
Synopsis: NFS attribute cache functions incorrectly
Release: 4.0, 3.4, 3.2

Description:
During periods of heavy NFS operations, attributes of a file in use with NFS server mount can get out of sync. An example of this problem may be seen when the following program is executed:

```
#include <sys/types.h>
#include <sys/stat.h>
#include <stdio.h>

static char *file = "/tmp/file";

main()
{
  struct stat  st;
  int    lasttime, fd;
  FILE  *fp;

  if (fork()) {
      if (stat(file, &st) {
            fprintf(stderr, "stat did not work - %d.\n", errno);
            abort(errno);
        }
      lasttime = st.st_mtime;
      for(;;) {
        if (stat(file, &st)) {
            fprintf(stderr, "stat did not work - %d.\n", errno);
            abort(errno);
        }
        if (st.st_mtime > lasttime) {
            printf("The file changed\n");
            lasttime = st.st_mtime;
        }
      }
  } else {
      sleep(1);
      fp = fopen(file, "w");
      fprintf(fp, "Here is line 1\n");
      fprintf(fp, "Here is line 2\n");
      fflush(fp);     /* not really necessary but ... */
      fclose(fp);
  }
  puts("I'm done!");
  exit(0);
}
```

```
machine% cc stat.c
machine% a.out
The file changed
I'm done!
^Cmachine%
machine% cat file
machine%
```

General

Reference Number: 1003255
Synopsis: Excessive packet activity on diskless client
Release: 3.2

Description:
ND based diskless systems show anomalous packet activity when they shouldn't.

Program

Reference Number: 1001371
Synopsis: yppasswd responds with "couldn't change passwd"
Release: 3.3, 3.2, 3.0, 3.0alpha

Description:
Using yppasswd to change the password in the yp database fails when the ascii file is not /etc/passwd, although /usr/etc/rpc.yppasswdd is set up correctly.
If you use /etc/passwd as the ascii file, it succeeds.

Reference Number: 1001379
Synopsis: Rlogin and username length
Release: 3.0

Description:
There seems to be several different ideas of the maximum username length incorporated in related utilities such as "rlogin" (or "rlogind"), "su", "login", etc.
It appears that rlogin/rlogind's idea is incorrect, although it could be one of the other utlities that is wrong. In any case, they should all be using the same parameters for maximum username length.

Reference Number: 1004770
Synopsis: rlogin propagates incorrect tty width
Release: 3.2

Description:
When the width of the terminal or window on the local system exceeds 127 characters, 'rlogin' incorrectly propagates the size of the window to the remote system.

Reference Number: 1004777
Synopsis: rpc.lockd core dumps
Release: 3.2

Description:
A given machine experiences problems where '/etc/rpc.lockd' disappears after a while, causing all programs using 'lockf' to hang.

Reference Number: 1003156
Synopsis: telnet from a Vax to the Sun fails
Release: 3.2

Description:
Telneting from a Vax(4.4 VMS) to a Sun running 3.2 will fail but telneting from the Sun to Vax will succeed. According to the customer:
1. login prompt is displayed
2. he specifies his login name and hits <CR>
3. the first <CR> is not interpreted consequently he hits <CR> for the second time.
4. the second <CR> is acknowledged as the passwd entry and the program exits with an invalid passwd entry message.

Work around:
Use 3.0 '/usr/etc/in.telnetd'.

Protocol

Reference Number: 1004409
Synopsis: telneting to a remote machine called "x" will fail
Release: 3.2

Description:
If a machine's hostname is specified as 'x', then an attempt is made to telnet to that machine, telnet fails and responds with the following error message:

```
% telnet x
Trying 0.0.0.0 ...
telnet: connect: Can't assign requested address
```

This happens because the Sun's internal routine to get the internet address takes a leading x to mean the address is given in hex, so 'x' is a valid IP number of zero.

Reference Number: 1004840
Synopsis: RSTAT(3R) always returns 0 in statstime.if_opackets
Release: 3.2

Description:
The RPC system call 'rstat(3R)' takes a pointer to a 'statstime' type structure ('rpcsvc/rstat.h') as one of its arguments and , fills the structure upon return. The structure contains several members, each of which is filled correctly, with the exception of the output packets member 'if_opackets'. No matter which host is called or how many times the call is made, the 'if_opackets' field always remains 0.

Reference Number: 1001411
Synopsis: ypinit does not copy user-defined yp databases to slave server
Release: 3.2alpha, 3.0
Description:
We have installed one of our own yp maps on our master server. When a new slave server was installed, "ypinit -s master_name" was done. All the system yp maps were copied over to the slave server. However, our own yp map was not.
Ypinit(8) man page states -
"A YP database on a slave server is set up by copying an existing database from a running server."
This implies that ALL maps from the server will be installed on the slave server.

Reference Number: 1002940
Synopsis: yppasswd fails on lexically similar logins
Release: 3.2, 3.0

Description:
Given a situation where two users have identical logins except that one login is longer than the other, if the longer login is above the shorter in the yp password file, the shorter one is not found by yppasswd, and thus, the yellow pages password cannot be changed.
Work around:
Put the shorter login ahead of the longer one in the password file.

Shell

Shell

Bourne Shell

Reference Number: 1001427
Synopsis: exec in bourne shell gives "no stack space"
Release: 3.2alpha, 3.0

Description:

If you exec a process that doesn't exist, then exec one that does, the following messages appear:

```
no space
no stack space
```

When you get out of the process, an error message appears such as one of the following:

```
no memory
sorry, pid was killed due to swap problems in swapout:
  no swap space for U area
pid killed due to swap problems in xalloc: no swap for text
```

C Shell

Reference Number: 1002770
Synopsis: csh dumps core if history reference in backquotes
Release: 3.2

Description:

The C shell dumps the core for history references enclosed in backquotes. For example:

```
% `\!$`
or
% echo `\!*`
```

Reference Number: 1004091
Synopsis: csh login expansion doesn't always work correctly
Release: 3.2

Description:

csh +login-name expansion does not always work correctly, and sometimes return a null. The problem appears most often within the following scenarios:

Run csh with 'filec' set
user must be in the YP passwd database
user cannot be in the local /etc/passwd file
do something like 'ls +user<ESC>', then try 'ls +user' again

Work around:

Issue the following sequence of commands:
unset filec
echo +<user> (replace <user> with the relevant login name)
set filec

Reference Number: 1004261
Synopsis: kill command bug "IOT trap (core dumped)"
Release: 3.2

Description:

Nested command substitutions attempted using alias cause a core dump, as in the following example:
alias psn ’set kj=‘psf \!*‘; echo $kj[1]; unset kj’
alias psk ’kill ‘psn \!*‘’
When run, the code attempts to evaluate the nested substitutions, notices that one of its invariants has been violated, and aborts, producing the following message:
IOT trap (core dumped)

Reference Number: 1004318
Synopsis: command line overflow in backquotes hangs shell.
Release: 3.4beta, 3.2

Description:

When using a backquote expansion in a command line within the C shell, any backquoted command that overflows the command buffer hangs the shell.

Work around:

Avoid using command substitutions that are likely to produce voluminous output.

Reference Number: 1004815
Synopsis: ~<user> functions erratically when expanded to an empty string
Release: 3.2

Description:

The C shell has intermittent problems when expanding ~<username> into an empty string.

SunView

SunView
Library

Reference Number: 1003168
Synopsis: pw_putcolormap( ) doesn't immediately update the colormap if window size is too large
Release: 3.2

Description:
When using a colormap of size 256, the routine pw_putcolormap( ) does not take effect until the mouse is moved out and back into the window. A colormap of size less than 256 seems to work as expected.

Work around:
At the time when pw_putcolormap( ) is called, use the routines win_grabio( ) and win_releaseio( ) which seem to reset the colormap.

Reference Number: 1003694
Synopsis: tty: GET/PUT selections dont work if not owner of /tmp/ttyselection
Release: 3.2, 3.0

Description:
If the file /tmp/ttyselection is not owned by the current user of suntools, the GET/PUT selection functions do not work correctly.

The PUT operation does not put the current selection in the tty window onto the shelf. Instead, the current contents of the /tmp/ttyselection file are 'put' onto the global shelf. The GET operation 'gets' the old selection which is incorrect from the user's point of view.

If /tmp/ttyselection is not owned by the current user of the window system, 'tty windows' do not update this file with the current selection, because the modes of the file prevent anyone other than the owner from writing to the file.

Note: 'suntools' does not remove the /tmp/ttyselection files when it exits. Hence, if one user logs out and another logs in, this problem appears.

Work around:
There are 3 possible workarounds:

1. Have 'suntools' clean up and remove /tmp/ttyselection when exiting.

2. Remove the need for a /tmp/ttyselection file from the tty window.

3. Make /tmp/ttyselectionwritable by the 'world'.

Reference Number: 1003706
Synopsis: horizontal scrollbar for panel doesn't work very well.
Release: 3.2

Description:
Left and right buttons are occasionally inoperative when attempting scrolling on a panel.
Work around:
Use the middle key when attempting horizontal scrolling.

Reference Number: 1003877
Synopsis: SunView problem reading from standard input in dbx/scripts
Release: 3.2, 3.0

Description:
When attempting to read from the standard input in SunView code which is being run under dbx or from a shell script, the signal TTIN is generated and tty input turned off. In the case of a shell script, there is an additional effect of the program hanging until it is manually killed.

Reference Number: 1004138
Synopsis: sunview: popup panels w/ scrollbars don't work properly
Release: 3.4beta, 3.2

Description:
A popup frame displayed via window_loop( ) having a panel containing a scrollbar does not work properly in Sun OS Release 3.2. The scrollbar appears, but clicking in the scrollbar region does not scroll the panel as it should. The cursor does not change, possibly indicating that the scrollbar is not getting any input events at all. This worked fine in Sun OS Release 3.0.

Reference Number: 1004251
Synopsis: interpose on scroll events behaves erratically
Release: 3.4beta, 3.2

Description:

An application attempts to adjust two (identical) scrollbars simultaneously on one SCROLL_REQUEST. The method of implementation is to 'interpose' on the scrollbars, and when the event is received on the first scrollbar, call 'scrollbar_scroll_to' on the second scrollbar with the 'view_start' obtained from a 'scrollbar_get' on the first.

The problem is only seen when using the page buttons. After a specific sequence of steps, the behavior on the second scrollbar shows no relation to the requests being made on the first.

Reference Number: 1004423
Synopsis: wmgr_figuretoolrect fails and trashes stack
Release: 3.4beta, 3.2, 3.0

Description:

In Release 3.2, calling 'wmgr_figuretoolrect' with a 'rect' whose values are set to WMGR_SETPOS causes overwriting of the stack.

In Release 3.4, the stack is correct, but the 'rect' values are not set.

Work around:

A user can workaround this problem via the following undocumented and unsupported method:

```
        /*    Declare a dummy rect   */

    Rect   dummy;

        /*    Replace call to wmgr_figuretoolrect()
         *    with call to wmgr_get_placeholders()
         *    with two arguments. First argument is the
         *    Rect that you would otherwise pass to
         *    wmgr_figuretoolrect(), second is the dummy
         *    Rect.
         */

    wmgr_get_placeholders(&rtool,&dummy);

        /*    Rect values in rtool are now set.      */
```

Reference Number: 1004512
Synopsis: Choice-item layout problem in panel package
Release: 3.2

Description:

Using MARK and NOMARK images for a choice item in the panel package and using the MARK_XS and MARK_YS attributes, the user attempts to control the placement of the MARK and NOMARK images in order to put the images exactly adjacent to each other. However, before drawing the 'mark' image the 'panel' package clears a 'rect' that is the size of the 'mark' image plus 3 pixels. It then draws the 'mark' image. Because I have two images adjacent to each other, this causes the first 3 pixels of the second image to get cleared.

Reference Number: 1004580
Synopsis: textsw_file_lines_visible( ) returns incorrect values
Release: 3.2

Description:

If a SunView application calls 'textsw_file_lines_visible (&top, &bottom)' when there is no text in a 'textsw', it returns the values 'top = 0','bottom = -1'. If the SunView application then reads a file into the 'textsw', and calls 'textsw_file_lines_visible( )' again, it still returns (0, -1).

Work around:

Scrolling or editing the text makes it return the correct values.

Reference Number: 1004860
Synopsis: user defined DEVID's incorrectly look negative
Release: 3.2

Description:

The file '/usr/include/sundev/vuid_event.h' states that 'Vuids' from '0x80' to '0xFF' are reserved for Sun customers.

If any of these vuid's are used, they create vuid's that start at '0x8000'. These get stored as 'signed shorts' in the 'Event structure' and get passed as 'signed shorts', which have their signs extended into negative integers, in such routines as 'win_post_id'.

If the programmer defines event codes in the same way that the system does in 'vuid_event.h', then the event codes get defined as 'signed integers', not as 'shorts' (the C compiler does not recognize constant integers of type 'short').

Thus, any event routine gets an event whose 'ie_code' is a negative number (such as 0xFFFF8000), and if the application tries to compare it against a positive integer (such as 0x000080000), it fails.

Work around:

Redefine 'event_id(e)', defined in /usr/include/sunwindow/win_input.h as follows:

```
((event)->ie_code)
     to
((unsigned short) (event)->ie_code)
```

Reference Number: 1004873
Synopsis: Some escape sequences crash TTY subwindows
Release: 3.4, 3.3, 3.2

Description:

If the user sends the following string (where '^L' means 'control-L' (FF) and '^[' means 'ESCAPE') to a 'tty' subwindow, the program providing the 'tty' subwindow goes away:

```
^L^[[0;0H^[[107030@
```

However, in some cases, the upper portion of the entire screen gets zebra-stripes painted on it. Infrequently, the system crashes because the 'window lock timeout' routine tries to do a 'printf' using the string in the 'lock' data structure. However, because that data structure has been zeroed out, the pointer it passes as a 'string' argument to 'printf' is NULL.

Work around:

Don't send escape sequences like '^[[107030@'. This particular sequence asks the 'tty' subwindow to insert 107,030 spaces.

Reference Number: 1004901
Synopsis: tty subwindow terminal emulator processes some ESC sequences wrong
Release: 3.2

Description:

When a customer uses the Sun as a terminal to another system, the following 'ESC' sequences do not work correctly in 'tty' subwindows, but run correctly outside of the window system:

1) ESC E[2H followed by
ESC E[3H followed by
ESC E[3,9H followed by
ESC E[3H

(originally noticed when the user typed '^N ^N ^E ^A' under 'EMACS')

2) ESC E[H followed by
ESC E[J

(originally noticed when the user typed '^L' under 'EMACS')

The incorrect behavior sometimes manifests itself as a failure of 'EMACS' to update the cursor position.

Reference Number: 1005083
Synopsis: tty: create and destroy loop of tty window eventually fails
Release: 3.4, 3.2, 3.0

Description:
If an application does a 'create' of a 'tty' window followed by a 'destroy' of that window several times (on the order of 3 to 10 times), the application eventually fails. This failure results in one or a combination of the following:

1. Prints 'WIN ioctl number C014671F: Bad file number' type error messages.

2. Prints the message 'couldn't find user name'.

3. Application dies with a segmentation failure.

These failures are indicative of partial memory trashing, perhaps due to incomplete cleanup by the destroy of the 'tty' subwindow.

Work around:
Instead of creating/destroying tty windows, one can repeatedly fork and exec shelltools. But this only works if you want a separate frame.

Reference Number: 1005102
Synopsis: storage leak in 3.4 sunview
Release: 3.4, 3.2

Description:
The panel_destroy_item( ) call does not always reclaim the storage that has been allocated to the panel item.

Reference Number: 1005173
Synopsis: Shelltools do not redraw emboldened text after being resized.
Release: 3.4, 3.2

Description:
When the user brings up 'shelltools' in emboldened mode and resizes it, the text is not redrawn.

Program

Reference Number: 1003138
Synopsis: csh filename completion inoperative in csh cmdtool
Release: 3.2

Description:
The ESC mechanism of csh, which provides fieldname completion, does not work with cmdtool in Release 3.2

Reference Number: 1004292
Synopsis: cmdtool core dumps if no permission to write to /tmp
Release: 3.4, 3.4beta, 3.2

Description:
In release 3.2, if the permissions of the '/tmp' directory deny the user 'write' access, 'cmdtool' dies with the following error message:

```
Memory fault - core dumped
```

Under the same conditions in release 3.4, the 'cmdtool' prints the following error message:

```
window: subwindow creation failed
```

However, the system creates the 'cmdtool' anyway.
Since 'cmdtool' is not able to save necessary information in /tmp, it should gracefully terminate.

Reference Number: 1004373
Synopsis: defaults edit does not update .indent.pro
Release: 3.4, 3.4beta, 3.2

Description:
The editing tool 'defaultsedit' does not update the '.indent.pro' file, even though system messages indicate that the file has been updated.
Work around:
Change .indent.pro manually.

Reference Number: 1004755
Synopsis: cmdtool strings larger variable size quit disappear
Release: 3.2

Description:
If a user enters a large string (over 2000 characters) without embedded 'carriage returns' into a 'cmdtool', the 'cmdtool' disappears and no error message is sent to the console window.

Reference Number: 1004882
Synopsis: Occasionally, put and get don't work correctly in textedit
Release: 3.2

Description:
Occasionally, the text that a user 'puts' on the shelf using the 'L6' key remains on the shelf when the user selects text in another window.

Reference Number: 1005033
Synopsis: F1 caps lock changes function key codes
Release: 3.2

Description:
The 'caps lock' feature on the 'F1' key changes the control codes sent by function keys. The normal printable characters of the control codes are usually in lower-case, but after selecting caps lock with the 'F1' key, these characters are now in upper case.

Reference Number: 1005369
Synopsis: gfxtool hangs input to other shelltools on Sun-4
Release: 3.4

Description:
When 'gfxtool' is in the window system along with one or more shelltools and keyboard input is typed into the shell of 'gfxtool' and then some keyboard input is typed into a 'shelltool', the text entered into the 'shelltool' does not appear immediately. Sometimes, if some time elapses, the shelltool's input may appear. Typing many carriage returns to the 'gfxtool' also frees up the shelltool's input.

Work around:
After typing commands in the 'gfxtool', there is a 'timeout' period, after which the 'shelltool' acts normally. Hence, the user can wait until the 'timeout' is up.

SunWindows

Reference Number: 1003720
Synopsis: pw_putattributes( ) does not set retained memory pixrect attributes
Release: 3.2

Description:
This is a memory pixrect bug. The pixwin function pw_putattributes( ) does not set the attributes of the retained memory pixrect (if present). This effectively prohibits the use of attributes on a retained canvas (including double buffering).

Reference Number: 1003864
Synopsis: crosshair cursor doesn't work if CANVAS_FAST_MONO is used
Release: 3.2

Description:
Crosshair cursors don't work in a "CANVAS_FAST_MONO" canvas on a desktop on a cg4 frame buffer. The crosshair cursor is not erased properly, and many repetitions of the following set of error messages appear on the console:

```
Kernel cursor Roperr 3
Kernel cursor Roperr 4
```

Reference Number: 1003908
Synopsis: Bad repainting from retained pixrect if pw_set_xy_offset used
Release: 3.2

Description:

An application calls 'pw_region' and 'pw_set_xy_offset' to set up a clip region in pixel coordinates. Thus, in a window of 200 x 200 pixels, the application could draw a 'polyline' from a coordinate list, then do the following:

```
newpw = pw_region(pw, 100, 100, 50, 50);
pw_set_xy_offset(newpw, 100, 100);
```

and redraw from the same coordinate list, drawing over the portion within the clip region. The application utilizes the 'pw_set_xy_offset' call to avoid having to adjust all the coordinates in the list.

This works correctly while drawing in the primary 'pixwin', but when the application calls 'pw_repairretained' to do damage repair, the image on the screen moves left and downward by (100,100) pixels.

Experimentation shows that 'pw_repairretained' ignores the offset. Attempts to set it to (0,0) before doing damage repair results in shifting the whole image over. The amount of shift respresents where the image would have been drawn in the pixwin in the offset had been (0,0) before drawing. This amount of shift also matches the part of the image that is misdrawn by 'pw_repairretained'.

System Administration

Install

Reference Number: 1003166
Synopsis: tapeless server installations occasionally fail in 3.2
Release: 3.2

Description:

During installation, the server crashes with a "dup ialloc" panic after installing the first client (3/75) when attempting to install the following configuration:
3/280 server, xylogics 451 controller, fujitsu disk
3 clients: 3/75, 3/140 and 2/50
3/110 tape server
However, when only one client was specified, no problems were seen. This system has been previously installed as a tapeless server running 3.2 with 2 dummy clients.
All clients were halted during the installation.

Reference Number: 1004202
Synopsis: can't install 3.3 on 3.2 EXPORT
Release: 3.3

Description:

The 3.3UPGRADE script fails to install on 3.2EXPORT. When the script checks /etc/motd for the string "3.2", it locates "3.2EXPORT" and responds as an incorrect release.

Work around:

Edit /etc/motd to have your release be "3.2" instead of "3.2EXPORT".

Setup

Reference Number: 1001496
Synopsis: setup
Release: 3.0

Description:

In situations involving Yellow Pages installation on a slave server from a remote tape on the master server (where the remote tape name and the master server name are the same, and the input device is a cursor-addressable terminal), when "Master Name" is entered, the message 'workstation name "xxxx" already in use' appears, and the user is not able to continue.

Work around:

Give a dummy name to the question:
'Tape Server Name :'.

Reference Number: 1001524
Synopsis: 3.0 setup doesn't display console error messages
Release: 3.0alpha

Description:

In the textsw in bitmap mode, 3.0 setup displays the exit status only--
the console error messages that should be written by
the 'printf' statements in the kernel do not appear.
The following is a sample message:

```
"command (/etc/ifconfig ec0 myserver -trailers up || /etc/ifconfig
 ie0 myserver -trailers up) > /dev/null 2>&1 exit status 1"
```

The problem is that command output is being directed to /dev/null,
and there is no way to tell what kind of problem caused the
ifconfig commands to exit abnormally.

Reference Number: 1001527
Synopsis: Mail configuration in setup makes no difference
Release: 3.0beta

Description:

During system setup, the mail server or client is designated. When
setup is complete, there is no change reflected in mail setup.

Reference Number: 1001528
Synopsis: Setup doesn't notify that it couldn't load tape
Release: 3.0alpha

Description:

Setup does not give an error message to show if loading of one
of the tapes was not successful. For example, when setup was
run and read the second tape containing /usr, it responded with
a message indicating tape exited with status 1 please insert tape 3,
but did not indicate that none of /usr was extracted.

Reference Number: 1001539
Synopsis: setup: does not install sendmail.cf on diskless clients
Release: 3.0beta

Description:

Setup for a server serving diskless clients installs the mail system on the server, but not for the diskless clients. On the server, setup copies either the /usr/lib/sendmail.main.cf file or the /usr/lib/sendmail.subsidiary.cf file to /private/usr/lib/sendmail.cf (specified by the link /usr/lib/sendmail.cf), depending on whether or not the system is setup as a main mail machine. Setup does not perform this copy for any of the diskless clients, which results in /usr/lib/sendmail.cf linking to nothing. sendmail will not be started up in /etc/rc.local without the /private/usr/lib/sendmail.cf file.

Reference Number: 1001542
Synopsis: Setup failed to properly install a yellow pages slave server
Release: 3.0

Description:

When running the 3.0 setup for a 68020, the selected setup option is a yellow pages slave server. The prompt responses are for the name of the master server (which is up at the time) and the full Internet address of the master server. After setup had finished extracting from all four tapes, a message is echoed, similar to the following:

```
making swagman a yellow pages slave server
```

The yellow pages databases are not copied over, and no error messages appear.

When the system is booted, ypserv runs, but it becomes apparent that the yellow pages databases had not been copied over.

Reference Number: 1001551
Synopsis: Setup won't use the same machine name for remote tape and yp master
Release: 3.2, 3.2alpha, 3.0

Description:

When installing setup from a remote host (the host is the yp master and the terminal being used is the yp slave server), setup does not accept the machine name the second time. The following error message appears:

```
Workstation name "name" is already in use
```

Work around:

Don't have setup setup the slave server, and go back in and do it by hand. That is, on the machine being installed, identify the machine as a yp client, and complete the installation. Then, do the following to make it a yp slave server.

run /usr/etc/yp/ypinit on the slave
add ypxfer requests to /usr/lib/crontab
go to the master servers and add the slave
to the ypservers map; directions in sysadmin manual.

Reference Number: 1002661
Synopsis: setup screen "clients" does not show all supported models
Release: 3.2, 3.2beta

Description:

The "clients" screen in setup does not show all the existing Sun systems. There is no card for models 100, 3/110, 3/260 or 3/280, and no easy-to-locate sources for root and swap size information for these systems in the installation documentation.

Work around:

Use any Sun-3 model for a Sun-3, and any Sun-2 for a Sun-2--they are all the same.

Reference Number: 1003096
Synopsis: setup rearranges disk partitions if a hole is present
Release: 3.1, 3.0

Description:

If a disk other than xy0 has a hole in it
(for example, "a,b,space,d,e"),
then setup automatically rearranges things to
look like "a,b,d,e,space", without receiving
any specific request from the user.

Reference Number: 1003159
Synopsis: Tapeless install only works with Class C addresses
Release: 3.0

Description:

When setup is performed as part of a tapeless installation, it is successful only when both the tape server and the client have Class C internet addresses. When setup is performed with Class A and/or B addresses, it does not operate correctly.

Utilities

Reference Number: 1004874
Synopsis: /etc/dump doesn't always return correct exit codes
Release: 3.2

Description:

'/etc/dump' does not always return the correct exit codes. For example, an exit code of 1 is described in the documentation to act as a normal exit, but there are some cases when 'dump' incorrectly returns 1. The errors seem to be caused by various routines in '/usr/src/etc/dump/dumprmt.c' that incorrectly 'exit(1)' when they probably should only 'return(1)'.

Reference Number: 1005619
Synopsis: Not all NFS clients consistently receive shutdown messages
Release: 3.4

Description:

When shutdown is run on an NFS server, clients do not reliably receive the shutdown message.

Work around:

Make '/etc/rmtab' have less information.

Reference Number: 1001565
Synopsis: MAKEDEV sfX
Release: 3.0

Description:

MAKEDEV sfX does a mknod with the wrong minor device numbers.
MAKEDEV sfX should also do a mknod for "sfpcX" devices.

The devices that should be created with a "MAKEDEV sf0" command are:

```
mknod sf0 b 9 0       # this is correct in MAKEDEV now
mknod sfpc0 c 33 0    # this was named "rsf" in old MAKEDEV
mknod rsf0 c 33 4     # this was 33 0 in old MAKEDEV
```

Utilities

Utilities
Editor

Reference Number: 1001724
Synopsis: vi wrapmargin set followed by '.' (repeat) produces errors

Description:
When 'wrapmargin' is set in vi, then text is inserted by repeating the last command using '.', any inserted text causing a line wrap does not appear as it should.

Reference Number: 1004858
Synopsis: vi :ta tags file does not correctly handle relative pathnames
Release: all

Description:
If a ':ta' tags file in 'vi' contains relative pathnames, 'vi' looks for those files relative to the current working directory, rather than relative to the directory in which the tags file resides.
Work around:
If source is available, add code to 'tagfind()' in the 'ex_cmdsub.c' to save the directory of the tags file. If the target file does not exist, prepend it to the target filename.

Formatter

Reference Number: 1001748
Synopsis: 'vtroff -F nonie -me' produces Floating Exceptions with .hl macro
Release: 3.0, 2.2, 2.0

Description:
'vtroff -F nonie -me' produces Floating Exceptions with .hl macro.
It seems to be actually caused by the '\l' troff command.
Work around:
Change the .hl macro definition as follows:

```
.de hl
.br
\l'\\n(.lu-\\n(.iu_'
.sp
..
```

The _ at the end of the \l argument
indicates to use the ASCII character _ instead of \(ru."

Reference Number: 1001757
Synopsis: man macro package - TH does not work as documented
Release: 1.2, 1.1

Description:

The .TH macro in the -man macro package does not function properly to set page, chapter, page foot center, page foot left, and page head center.

Reference Number: 1002807
Synopsis: .TH macro of -man package defective
Release: 3.2alpha, 3.0

Description:

The bug can be seen by placing the following line in a nroff file (nroffed with the -man option):

```
.TH u2dm 1D "14 July 1986" "XYZ Corporation"  "DM8000 Commands"
```

The following is output as the header:

```
u2dm(1D)    UNKNOWN SECTION OF THE MANUAL    u2dm(1D)
```

The following is output as the footer:

```
Sun Release 3.0B   Last change: 14 July 1986          1
```

Work around:

If the following changes are made to /usr/lib/tmac/tmac.an, it conforms more closely with the doumentation in man(7).

output from diff -c on /usr/lib/tmac/tmac.an

```
***************
*** 112,119
  .if t .po .588i
  .ll 6.5i
  .nr LL \\n(.l
- '    # this seemed to be missing -- pZ XYZ
- .ds ]W \\$4
  .ds ]H \\$1\|(\|\\$2\|)
  .ds ]D UNKNOWN SECTION OF THE MANUAL
  .if '\\$2'1C' .ds ]D USER COMMANDS

--- 112,117 -----
  .if t .po .588i
  .ll 6.5i
```

```
 .nr LL \\n(.l
 .ds ]H \\$1\|(\|\\$2\|)
 .ds ]D UNKNOWN SECTION OF THE MANUAL
 .if '\\$2'1C' .ds ]D USER COMMANDS
***************
*** 143,150
 .if '\\$2'8'.ds ]D MAINTENANCE COMMANDS
 .if '\\$2'8C' .ds ]D MAINTENANCE COMMANDS
 .if '\\$2'8S' .ds ]D MAINTENANCE COMMANDS
-'    # this also seems to be missing -- pZ XYZ
- .if !'\\$5'' .ds ]D \\$5
 .wh 0 }H
 .if t .wh -1i }F
 .if n .wh -1.167i }F

--- 141,146 -----
 .if '\\$2'8' .ds ]D MAINTENANCE COMMANDS
 .if '\\$2'8C' .ds ]D MAINTENANCE COMMANDS
 .if '\\$2'8S' .ds ]D MAINTENANCE COMMANDS
 .wh0 }H
 .if t .wh -1i }F
 .if n .wh -1.167i }F
```

Mail

Reference Number: 1001798
Synopsis: A small mail message sent to a long alias fails
Release: 3.0

Description:
When a small mail message (eight or fewer characters) is sent to a long alias (eleven or more recipients), the message "No message body" appears and mail terminates. If the message size is increased or the number of recipients in the alias is decreased, the problem disappears.

Reference Number: 1003370
Synopsis: Mail file command gets confused by cd command

Description:
When "file" is used to access mailbox in another directory, then "cd" is used to change to that directory to save messages there, the second "file" command produces an error message. "file" tries to write out the file that was opened previously, but it doesn't keep the pathname to that file, and tries to modify a file relative to the current working directory. "quit" also fails.

Reference Number: 1003400
Synopsis: Mail, v command doesn't set current message
Release: 3.2

Description:
When supplying a message number with "v", Mail doesn't set the current message to that number (unlike other commands that do).

Make

Reference Number: 1003151
Synopsis: make does not always build objects that it should
Release: 3.2

Description:
The 3.2 version of make does not build specified objects when those objects depend on another object that has no build lines in that make file. This occurs with both pre- and post-3.2 make on remote files, when the server and client don't agree on the time.

Reference Number: 1003818
Synopsis: make -f ./file fails, but make -f file succeeds
Release: 3.2

Description:
"makefile -f ./file" responds that it cannot build a file from implicit rules, but "makefile -f file" can.
Work around:
Apparently, make filenames that start with . are trouble, so don't use them.

Reference Number: 1002641
Synopsis: make does an incorrect sccs get on (Makefile) include files
Release: 3.2, 3.0

Description:
Make does not do an "sccs get" correctly on Makefile include files. The get is done to the sccs directory, instead of to the proper
.if n .wh -1.167i }F

Printer

Reference Number: 1004022
Synopsis: lprm checks for super-user strangely
Release: 3.2

Description:
lprm will not allow a user who used "su" to access root to remove other users' files; the user must login as root. lprm checks for super user privileges by comparing the "getlogname" with "root".
Work around:
rlogin to "root" or login as "root"

Program

Reference Number: 1001915
Synopsis: script(1) does not output parity thru tty port
Release: 2.0beta, 1.1

Description:
When using ASCII terminals requiring even or odd parity (that is, terminals that are unable to ignore incoming parity, and must be set to even or odd) with script(1), characters that need their eighth bit set to '1' are not accepted by the terminal because they are sent with this bit as '0'. This is the result of script putting the tty into raw mode, and also applies to other programs that put their tty into raw.
Work around:
Manually set the drivers mode to cbreak instead of raw.

Reference Number: 1002024
Synopsis: problem with quota
Release: 3.0

Description:
When /bin/login runs /usr/ucb/quota on a Sun 3/75 running 3.0 to login, a delay of 4-5 minutes occurs. The kernel used was built from SDST160 and does not have QUOTA support. There are no NFS filesystems mounted with the quota mount options, and some are explicitly turned off. Some servers are 2.x machines lacking the rpc.rquotad daemon. The /usr/ucb/quota code doesn't check to see that the local filesystems are mounted noquota.

Reference Number: 1002039
Synopsis: INDXBIB causes memory fault message and dumps core.
Release: 3.0, 3.0alpha

Description:
INDXBIB causes memory fault message and dumps core. /usr/lib/refer/inv, which is called by the indxbib shell script, is the component which produces the memory fault message.

Reference Number: 1002077
Synopsis: Vi invoked from makefile aborts on ctrl/c
Release: 3.2beta

Description:
when vi is invoked from inside a makefile, then ctrl/c is typed (inside vi), make aborts, leaving the screen in vi.

Reference Number: 1002085
Synopsis: bc incorrect in hex fractions
Release: 3.0

Description:
"bc" calculator produces incorrect results when using hex fractions.
Work around:
Accuracy can be controlled by zero padding the fractional value after the decimal point.

Reference Number: 1003481
Synopsis: grep -i does not work properly
Release: 3.2

Description:
"grep -i" does not properly match strings having an uppercase character after a closure.

Reference Number: 1002095
Synopsis: Catman problems.
Release: 3.2alpha

Description:

When running catman, the following error messages appeared:
opendir: mann: No such file or directory
.: No such file or directory

Reference Number: 1002114
Synopsis: lookbib doesn't find anything unless indxbib run
Release: 3.2alpha, 2.3

Description:

"lookbib" does not find anything unless "indxbib" is run. "lookbib" should create an foo.ig file, then perform an "fgrep" on the file to find the string. The foo.ig file is created, but nothing is found.

Reference Number: 1002600
Synopsis: pg always uses default window size
Release: 3.2beta

Description:

"/usr/5bin/pg" does not fill in a window larger than the default window size, and overfills a window that is smaller than the default window size.

Reference Number: 1005119
Synopsis: Flow-control problem when tip is run from tektool
Release: 3.2

Description:

When an application uses 'tip(1)' from 'tektool(1)' to view displays generated by Pyramid hardware, the resulting display shows various errors.

uucp

Reference Number: 1002195
Synopsis: timeout for uuxqt lock files is too short
Release: 2.0

Description:
Because uuxqt has a one-hour timeout on its LCK.XQT lockfile, uuxqt ignores the lockfile when a long news job is running (and the fact that the job is still running) and restarts the job.

Reference Number: 1003563
Synopsis: tip doesn't renew uucp lock files
Release: 3.2

Description:
'tip' doesn't update the change-time on a 'uucp' lock file. 'uucico' ignores any lock file older than 1.5 hours by re-using the line at the end of that time. Additionally, 'tip' does not know that a lock file that has not been changed in the past 1.5 hours is out of date.
Work around:
Do not use 'tip' on a system running 'uucico', or don't use 'tip' connection for more than 1.5 hours.

Reference Number: 1002196
Synopsis: uucp Hayes modem support doesn't work as documented
Release: 2.2, 2.0

Description:
When trying to override the auto-answer feature Sun's uucp sets for Hayes modems, the instructions in the 2.0 Release Manual (at the top of page 12) indicate it is possible to issue Hayes 'Set' commands by prefixing the phone number in the L.sys file with a '-', but this doesn't work.
Work around:
Exchange the documented '-' for the Hayes ';' command. This puts the modem back into command state after dialing, permitting the software to send the Hayes 'Set' command, and setting register 0 to 0.

Reference Number: 1002903
Synopsis: uuname can not handle long L.sys entries
Release: 3.2, 3.0

Description:

If given a very long L.sys line, uuname responds with spurious host names that include part of the long L.sys line (possibly a security breach). This is because uuname has a hard-coded character buffer for storing these things in, and truncates the line to that size without flushing the rest of the entry.

Work around:

Keep L.sys entries short, if possible.

Reference Number: 1003443
Synopsis: uucp and uux allow only 19 fields on an L.sys line
Release: 3.0

Description:

uucp and uux allow only 19 fields on an L.sys line. If there are more fields, uucp and uux usually respond with the following message:
"bad system name"
uucp and uux can also fail randomly, and very complicated logins confuse them.

The problem is found in versys.c, where getargs is used to fill in a 20-word array of character pointers.

Reference Number: 1003870
Synopsis: uucico security leak
Release: 3.2, 3.0

Description:

When "cd /usr/lib/uucp" is entered, then "uucico -r1 -x5 -s<uucp_contact_system> &", uucico tries to connect to <uucp_contact_system>. If it succeeds, it responds with the password.

Work around:

chmod o-rwx /usr/lib/uucp

Other Sun Software Products

SunAlis
Database

Reference Number: 1004494
Synopsis: Can't create database from filing menu without using
model document
Release: SunAlis2.0

Description:
User can't create a database from the filing menu without specifying a model document.

Documentation

Reference Number: 1005103
Synopsis: Method to get Alis mail out of Unix mail is not documented.
Release: SunAlis2.0

Description:
The SunAlis manual omits how to send UNIX mail to SunAlis mail.
Work around:
Edit the file 'acapps' in the SunAlis executables directory, such that the entry for the 'gtwy3' process has a 'y' instead of a 'n'. This will start the 'gtwy3' process with 'aop', and SunAlis mail will be extracted from UNIX mail into SunAlis as appropriate. (Also follow the rest of the instructions for setting up SunAlis to SunAlis mail that travels via UNIX with the gateway registry using 'gtwy2'.)

General

Reference Number: 1004096
Synopsis: SunAlis 2.0beta will not run on high-res monitors
Release: SunAlis2.0beta

Reference Number: 1004097
Synopsis: Alis garbles screen on gray scale monitors
Release: SunAlis2.0beta

Description:
When SunAlis is run on a raw gray scale monitor (ie., without suntools), some screens are garbled. This problem does not occur if SunAlis is started from a shelltool.

Reference Number: 1004182
Synopsis: alisverify segfaults when run by root on an nfs mounted fs
Release: 2.0beta

Description:
If SunAlis is installed via a nfs mount and you run 'alisverify -v' as root, a segfault occurs. If you run 'alisverify' as alis, it works, but gives the message that you need to run 'alisverify' as root to check the network.

Reference Number: 1005016
Synopsis: Problems with Alis and suntools windows
Release: SunAlis2.0

Description:
Resizing Sunview windows while running SunAlis may cause loss of parts of the SunAlis display or may cause SunAlis to crash.
Work around:
Avoid resizing Sun windows while running SunAlis.

Spreadsheet

Reference Number: 1000020
Synopsis: you can't protect formulas from being pasted over
Release: 3.0

Description:
It is not possible to protect formulas from being pasted over. When formulas are protected, the user cannot edit the area, but can paste right over it. This does not occur with regular cells.

Reference Number: 1004489
Synopsis: Spreadsheet column width > window width causes Alis to hang.
Release: SunAlis2.0

Description:

Using a spreadsheet column width greater than the window width will cause SunAlis to hang when user tries to access that column.

Datacomm

Datacomm

bsc3270

Reference Number: 1003789
Synopsis: pe3287 dies with panic(3) when run with bsc3270
Release: SunLink3.0

Description:
Users trying to send output to printers that they have defined in the BSC 3270 environment will not succeed. pe3287 dies with a panic code of 3 because its buffer size is too small.
Work around:
Patch tape available from Technical Support.

Reference Number: 1003791
Synopsis: pe3287 dies with panic(839) with WSF
Release: SunLink3.0

Description:
In the VM environment, occasionally a WSF gets sent to the pe3287, causing pe3287 to do a panic(839).

bscrje

Reference Number: 1000305
Synopsis: Data lost on records > 132 chars
Release: SunLink1.0

Description:
Data is lost in records exceeding 132 characters long when files are received in non-transparent 3780 mode and utilize space compression. A line feed character replaces the character at position 132 in each compressed record. Standard 3780 protocol restricts records to a maximum length of 144 bytes.

Reference Number: 1000307
Synopsis: records > 80 chars not accepted in 2780 transparent mode
Release: SunLink1.0

Description:
The gateway does not accept a file received in 2780 transparent mode with record lengths exceeding 80 bytes. An incorrect error

message is displayed, including all or part of the previous error message.

Reference Number: 1000308
Synopsis: line field in punch file with records <80 bytes returns error msg
Release: SunLink1.0

Description:
A file containing a line feed character is created when a punch file containing blocked records with some record lengths less than 80 bytes is received in 2780 transparent mode. This punch file is accompanied by the following error message:

PUN FAILED TO RECEIVE

Reference Number: 1000310
Synopsis: blocked records < 80 bytes in 2780 transparent mode cause error
Release: SunLink1.0

Description:
The gateway does not accept blocked records when some record lengths less than 80 bytes are received in 2780 transparent mode and with no component selection. The following error message appears upon receipt of such a file from Wang VS, referring to a console message rather than a printer file:

KEYBOARD FAILED TO RECEIVE- INVALID BLOCK/RECORD FORMAT

No file was created.

Reference Number: 1000319
Synopsis: Gateway does not time out a connection if waiting for data frame
Release: SunLink1.0

Description:
If the gateway is waiting for a data frame, it will not timeout if the sender goes away. Normally, DSR would tell us that the sender had terminated, but we cannot monitor DSR in synchronous mode. The driver and the state tables should be changed to set a long (-to) timer value and disconnect if the timer expires.

Reference Number: 1000321
Synopsis: Device/file name incorrect on some error messages
Release: SunLink1.0

Description:
The device name such as 'KEYBOARD', 'RDR', 'PTR', or 'PUN', and/or the filename are occasionally incorrectly reported in the error message for the previous file sent or received.

Reference Number: 1000322
Synopsis: Error message "Formatting error" reported twice
Release: SunLink1.0

Description:
When a file sent in 2780 or 3780 mode contains invalid control characters, the following error message is reported twice:

...formatting error...

Reference Number: 1000323
Synopsis: two gateways with the same name can cause confusing errors
Release: SunLink1.0

Description:
Starting up two gateways with the same name can cause confusing errors. If the system administrator starts two rjebsc programs with the same "name", the rjemapper process will remember the second one. However, the first one is probably the one that will successfully start up. The rje user will get "gateway not responding" if he tries to use the rje command because it will try to communicate with the second instance. This problem could be minimized by moving the code that registers the process with the rjemapper further down in the startup code (preferably after the process tries to open /dev/bsc*).

Reference Number: 1000324
Synopsis: Baud rates below 9600 baud cannot be set with syncinit
Release: SunLink1.0

Description:
Baud rates below 9600 baud cannot be set with syncinit. Syncinit cannot set the baud rate below 9600 due to a bug in zs_bsc.c. The variable tconst should be an unsigned short.

Reference Number: 1000325
Synopsis: HASP compression will put in extra char
Release: SunLink1.0

Description:
In the HASP compression mode, 'rjebsc' will put in an extra character if the last character is a duplicate character.
e.g., 'go to 100' becomes 'go to 1000'.

Reference Number: 1000333
Synopsis: 130+ lines mishandled in 2780/3780
Release: SunLink1.0

Description:
The gateway apparently allows print lines to have a maximum length of 132 characters, and attempts to segment longer records into 132-character increments. However, when print records longer than 130 characters are non-transparently received in 3780 or 2780 mode, the records are split after the 130th character, rather than the 132nd character. This seems to occur because the two-character vertical forms control sequence appearing at the beginning of the record is included in the character count.

Reference Number: 1000337
Synopsis: Received records = 132 characters are not treated correctly.
Release: SunLink1.0

Description:
For records received via 3780 mode, extra linefeeds are incorrectly inserted in received records containing 132 characters. Even records with compressed blanks have linefeeds incorrectly inserted after decompression to 132 characters.

This problem does not occur with records received via 2780 mode; the records containing horizontal tabs, when expanded to 132 characters, do not contain extra linefeeds.

Reference Number: 1003654
Synopsis: HASP: back-to-back operator commands may cause dump
Release: SunLink3.0

Description:
If you submit two "keyboard" commands so quickly that the second is queued before the first was sent, the second keyboard command is lost. An error message sometimes appears, and in some cases, the 'rjebsc' process dies with an address fault.
Work around:
Only submit one operator command at a time and wait for any host response before sending next command.

Reference Number: 1005029
Synopsis: rjebsc not fully initializing arg_table
Release: SunLink3.0

Description:
If the rjebsc table named 'arg_table[]' in 'parstab.h' contains 30 entries in the table, and the 'cmdta.h' file has 'TOT_ARGS = 27' as the number of arguments in the table, when the 'arg_table' is initialized with 'TOT_ARGS' number of entries, the last 3 entries are not initialized as 0. When the user enters the following:

```
/usr/SunLink/rjebsc/rjebsc rmt28 HASP /dev/dcpa3 -dy -l logfile -u&
```

The following error message is returned:

```
ERROR: ARGUMENT -u REPEATED
```

This error message is the result of the last three entries of the table not being 0.
Work around:
Make 'TOT_ARGS' equal the length of 'arg_table' in entries.

dna

Reference Number: 1003814
Synopsis: When a server reboots, all diskless clients running DNA hang
Release: SunLink4.0

Description:
If a server that is serving diskless clients running SunLink DNA crashes or is halted/rebooted in any way, the diskless clients running DNA will hang.

Work around:

Edit the /etc/rc.local file to invoke the following script at boot time after the nd command is executed.

```
#!/bin/csh -f
#
# JFC 1/9/87;   Fix the arp table on a server that has diskless
#               clients running SunLink DNA so that the DNA clients
#                  will not hang after the nd command is invoked.
#
set clientlist=(`awk '/^user [A-Za-z][A-Za-z0-9]* 0/ { print $2 }'/etc/nd.local`)
foreach i ($clientlist)
/etc/arp -d $i > /dev/null
end
```

Reference Number: 1003840
Synopsis: dnalogin doesn't handle the backspace character correctly
Release: SunLink4.0

Description:

When remotely logged in to a DEC VMS systems, dnalogin does not handle backspace characters, arrow keys, ^A and other control characters correctly. This is not a problem when using dnalogin to login to another Sun.

Reference Number: 1003841
Synopsis: dnalogind doesn't handle multiple control characters correctly
Release: SunLink4.0

Description:

dnalogind does not handle muliple occurences of CTRL-C, CTRL-D, CTRL-Y, and CTRL-\ correctly. When remotely logged in to a Sun system from a VAX/VMS system using SET HOST, typing CTRL-C twice results in unexpected termination of the SET HOST session. Usually, no error message is given and the dnalogind process on the Sun is not terminated.

Reference Number: 1004426
Synopsis: Can't build DNA kernels on SunOS 3.3
Release: SunLink4.0

Description:
If you try to install SunLink DNA 4.0 on a SunOS 3.3 system, you will get a load error when you try to build a DNA kernel. The problem is that the routine atoi (_atoi) is multiply defined.

Reference Number: 1004429
Synopsis: dnalogin fails with segmentation violation
Release: SunLink 4.0

Description:
The bug occurs after using dnalogin to log in to VMS/VAX in SunLink 4.0. When the VMS/VAX system is running MONITOR PROCESS/TOPCPU and a MESSAGE is sent to the window from the VAX (such as when VAXmail arrives or when a job that was submitted to a queue with the /NOTIFY switch completes), the following error message is printed on the window and the VMS login session is terminated:

```
Fatal Unix Error (Segmentation Violation)
CTERM exiting
```

This happens both in vt100tool and shelltool when logged in using dnalogin. It happens in neither vt100tool nor shelltool when logged in via TELNET.

Reference Number: 1004432
Synopsis: multiple passwords doesn't work
Release: SunLink4.0

Description:
If the VMS system turns off echo, and then reads multiple lines of input with prompts between them, the vt100 window overwrites the prompt each time, rather than using a second line.

Examples of this are seen when changing the password, or logging in when the VAX has the option for TWO system passwords to be typed It 'appears' not to prompt for the second one, but in fact it does.

Reference Number: 1004475
Synopsis: writing 100,000+ bytes over /dev/dna will panic the system
Release: SunLink4.0

Description:
Writing 100,000+ bytes over a logical link causes the system to panic with the following message:

```
PANIC: CEX fatal DNA error
```

Work around:
Limit the maximum number of bytes done on each write to less than 32767 bytes. This a short word. vs. long word problem.

Reference Number: 1004716
Synopsis: dnalogin dies with protocol error 30
Release: SunLink4.0

Description:
When a user logs in to a VAX/VMS environment using 'vt100tool' and 'dnalogin', runs a SunINGRES application in one window, then sends mail to himself from the second window, the following message appears in the first window:

```
New mail arrived
```

When the user attempts to repaint the screen using '^W', 'dnalogin' dies with protocol error code 30.

Reference Number: 1005036
Synopsis: dnalogin doesn't handle raw mode properly
Release: SunLink4.0

Description:
When logged into VMS/VAX using 'dnalogin', an attempt to set the VMS tty to 'PASTHRU' (similar to 'RAW' mode) does not work properly. In 'PASTHRU', '^C' and '^Y' are normally ignored, but when dnalogged into VMS/VAX, '^C' and '^Y" are not ignored.

Reference Number: 1005044
Synopsis: ^D ^F ^E ^H ^J ^X not mapped properly
Release: SunLink4.0

Description:
The UNIX line editing functions '^D', '^F', '^E', '^H', '^J', and '^X' do not get mapped correctly from UNIX to VMS. For example, the UNIX '^W' function acts as a VMS '^J' function; the UNIX '^H' function acts as a VMS 'DELETE', but the cursor does not move; and the UNIX '^X' function acts as a VMS '^U' function.

sna3270

Reference Number: 1003690
Synopsis: cannot start more than 4 sna3274 sessions
Release: SunLink4.0

Description:
No more than 4 sna3274 sessions can be started on the same sna gateway. When you try to start the 5th cluster controller process, you get the error message:
<time stamp> 3270 server: register with mapper failed
The configuration used is:
3.0 unix, 4.0 sna3270, 2 SCP boards
The original 4 sessions were started on scp rs232 ports, and the fifth one was attempted on an scp rs449 port.
The /etc/servers file contains:

```
rpc    udp    /usr/SunLink/sna3270/3270mapper   100013      2
```

Only 1 3270mapper is running (as shown by ps ax)

Work around:
Patch tape available from Technical Support.

Reference Number: 1003792
Synopsis: lu1 scs pe3287 does not tab properly
Release: SunLink4.0

Description:
When using a SCS-type (lu1) pe3287 printer, tabs are not handled properly in 'f1scs.c'.

Reference Number: 1003794
Synopsis: lu1 pe3287 puts 2 returns after 132-col output
Release: SunLink4.0

Description:
When using a SCS-type (lu1) pe3287 printer, two carriage returns are inserted after 132-column output lines.

Reference Number: 1003795
Synopsis: lu3 pe3287 loses blanks in column 1
Release: SunLink4.0

Description:
When using a DSC-type (lu3) pe3287 printer, blanks are lost in the output of column 1.

Reference Number: 1003807
Synopsis: line still hung after re-dialing following a FRMR
Release: SunLink4.0

Description:
When running through an SCP at 4800 baud rate and some line noise is experienced, the SDLC code misinterprets the frame and sends out an FRMR. When the host sees the FRMR, it does not attempt recovery. The primary side is responsible for recovery, so the host appears to stop dealing with the link. SNA/SDLC must be stopped then restarted before redialing the host.

Reference Number: 1004758
Synopsis: sna3270 sends +rsp to WSFRPQ but mishandles the query reply
Release: SunLink4.0

Description:
When a Sun is connected to a Stratus host, the primary sends the Sun a 'Read Partition Query Write Structured Field' command. The Sun sends a '+rsp', even though the Sun does not support structured field commands. The query reply that follows the '+rsp' appears to be ill-formed and incorrect. This implies that the Sun responds incorrectly to all structured field requests.

vt100 emulation

Reference Number: 1004435
Synopsis: bugs using DEC package wps (word processing) with vt100tool
Release: SunLink4.0

Description:

When using vt100tool and loading wps, a DEC word processing package for vt100's, wps fills in a menu, and then fills in information such as folder names in reverse video. At the end of this sequence the cursor should be near the bottom of the screen in preparation for user input. However, the cursor is left at the end of the reverse video text.

Another problem with using wps on the vt100tool is that the exit screen (PF1) appears to generate the correct sequence, but wps doesn't respond to it.

X.25

Reference Number: 1004035
Synopsis: x29 does not return pad prompt from remote
Release: SunLink4.0

Description:

When running 'pad' on another machine using 'rsh', '^P' does not return the 'pad>' prompt.

Reference Number: 1004037
Synopsis: x29 pad creates serverlog with setuid root and execute permissions
Release: SunLink4.0

Description:

If the first user to run 'pad' after a system boot is 'root', the '/tmp/x29serverlog' file is created with 'setuid root' and 'execute' privileges, thus causing a security breach. The user is not given 'all' write permission to allow other users to append to the file.

Reference Number: 1004038
Synopsis: x29 program creates serverlog which is also setuid
Release: SunLink4.0

Description:
The 'x29' program creates a 'x29serverlog' file which has 'setuid' privileges, causing a security breach.

Reference Number: 1004083
Synopsis: x25start will not run from rc.local
Release: SunLink4.0

Description:
Because '/bin' is not in the 'PATH' for the '/usr/SunLink/x25/x25start' shell script, when this script is invoked from '/etc/rc.local', it does not run.
Work around:
Add the following to the 'PATH=' line in 'x25start' before '$PATH':

```
:/bin
```

Reference Number: 1004237
Synopsis: No mention of X.25 'sgttyb' definition moved to 'ioctl.h' in 3.2
Release: SunLink4.0

Description:
When loading X.25, the include file 'ioctl.h' is overwritten. In UNIX Release 3.0, the 'sgttyb' definition is in the include file 'sgtty.h'. In UNIX Release 3.2, this definition has been moved to the include file 'ioctl.h', but the version of 'ioctl.h' shipped with X.25 does not include this definition, or refer to this new location.

Reference Number: 1004645
Synopsis: HDLC does not recover from FRMR properly
Release: SunLink4.0

Description:
When using HDLC to transfer large (500K to 5Mb) files between two Sun systems using the sio ports, periodically the connection is lost and the following error message appears:

```
HDLC_FRMRIN[0]:ctl=91 ns=3 nr=6 resp=0 why=W
```

where 'ctl', 'ns', and 'nr' can be varying values, and 'W' is an indication that a Frame error has occurred with an invalid control field. HDLC then resets the virtual circuit to X.25 specifications upon receipt of an FRMR. Writes being issued to HDLC to send the data continue as if nothing is wrong. More than sixteen 1K buffers of data read into the HDLC driver are subsequently discarded. The Frame Rejects are being caused by mishandling of REJ packets.

SunGKS

SunGKS
Library

Reference Number: 1004504
Synopsis: C-binding ginpk (initialize pick) incorrectly calls init_pik
Release: gks2.0beta

Description:
In ginpk (initialize pick), a Gpick* input named pinit is passed to init_pik cast to a Pick*, as shown below:

```
gkstype.h line 685:
   typedef struct  /* pick data */
        {
        Gpstat  status;      /* pick status */
        Gseg    seg;         /* pick segment */
        Gpickid pickid;      /* pick identifier */
        } Gpick;

ugkstype.h line 395:
   typedef struct
           {             /* Pick input record: */
           Segname pik_seg;  /* picked segment */
           Pickid  pik_pid;  /* pick identifier */
           } Pick;
```

The following error message is returned:

```
Error 152 in function: Initialize Pick
Initial value is invalid.
```

Work around:
A call can be made directly to Prior's internal "init_pik".

Reference Number: 1004891
Synopsis: specifying error file in gopengks( ) fails to collect errors
Release: gks2.0beta

Description:
If 'gopengks' is called with an actual file as the error message file, rather than 'stdout' or 'stderr', and errors occur during runtime, the errors are not printed, nor are they collected in the specified file.

Work around:

Add the following line after the call to fopen(),
where errfile is the (FILE *) variable passed to 'gopengks':

```
setbuf(errfile,(char *)NULL);
```

Reference Number: 1004918
Synopsis: GKS window fails to redisplay properly after going FullScreen
Release: gks2.0beta

Description:

When a GKS window containing an image that is displayed and constantly updated goes 'FullScreen' as a result of the user selecting the 'FullScreen' option from the frame menu, the image intermittently fails to be properly redisplayed. The new images are drawn, but the old images remain on the screen. The old images can be erased by selecting the 'Redisplay' option from the frame menu.

Reference Number: 1004997
Synopsis: gqlgdp returns garcgdp, gcirgdp, grecgdp as undefined symbols
Release: gks2.0beta

Description:

When the 'C INQUIRE LIST OF AVAILABLE GDP' (Generalized drawing primitives) function is called using either the 'ginqavailgdp' long name or the 'gqlgdp' short name, the following symbols are specified as undefined:

```
agqlgdp_sw-O.o:     U _garcgdp
agqlgdp_sw-O.o:     U _gcirgdp
agqlgdp_sw-O.o:     U _grecgdp
```

Reference Number: 1005287
Synopsis: segment information incorrectly recorded in GKS segment state list
Release: gks2.0

Description:

The 'workstation state' list and 'gks segment state' list incorrectly record segment information. This happens after a particular sequence of workstation and segment operations have been performed on a sun_tool and a metafile output workstation.

Reference Number: 1005582
Synopsis: Fill area ignores pattern reference point
Release: gks2.0

Description:

If a fill area is done before a set pattern reference point, any fill areas following the set pattern reference point use (0,0) as the reference point, rather than the value set.

Work around:

Setting the pattern reference point before calling fill area will work if only one pattern reference point is used.

Reference Number: 1005615
Synopsis: Inqpixelarray does not work on the 'tall' sun_tool workstation
Release: gks2.0

Description:

'inqpixel' and 'inqpixelarray' functions work properly only on the square-shaped or wide sun_tool workstations. Both functions fail when used on the tall workstation, where the height is greater then the width.

Reference Number: 1005627
Synopsis: Clearing display surface does not occur on sun_canvas workstation
Release: gks2.0

Description:

Clearing display surface does not occur within delete segment, segment transformation, redraw all segment on workstation, or clear workstation operations involve a display refresh on a sun_canvas workstation. The user must execute the frame menu 'redisplay all' option to update the display.

SunINGRES

Documentation

Reference Number: 1000951
Synopsis: no doc on enviroment variables ING_BACKEND and ING_PRINT
Release: SunINGRES 2.0

Description:
Additional release information:
23se19 Installation Guide Rev C (Aug 21 84)
There is no documentation about the enviroment variables
ING_PRINT and ING_BACKEND.

Reference Number: 1000956
Synopsis: left out single quotes in documentation for report command
Release: SunINGRES 3.0

Description:
On page 4-5 of the 3.0 Report Writer Reference manual it says under examples:

```
report mydb myrep (sal=10000,dept=toy)
```

If you try this command it will complain about badly placed parentheses.

Manual should say the following:

```
    report mydb myrep '(sal=10000,dept=toy)'
```

Reference Number: 1002666
Synopsis: C variables in an equel program are limited to 12 characters
Release: SunINGRES 3.0/25

Description:
The Equel manual should state that declaring C variables longer then 12 characters in 'eqc' produces faulty C code.

Reference Number: 1004121
Synopsis: equel documentation says iiseterr and it should be IIseterr
Release: SunINGRES 3.0/25

Description:
The equel/C manual (p. 3-3 to 3-5) discusses the Ingres routine iiseterr. This is incorrect. It should be IIseterr.

Reference Number: 1004487
Synopsis: Section 7.6 of Ingres/EQUEL/C User's Guide is incorrect
Release: SunINGRES 5.0

Description:
The 'cc' line in section 7.6, p. 7-6 of the SunINGRES/EQUEL/C User's Guide is incorrect. The line currently reads as follows:

```
cc -I ingres/lib filename
```

The 'cc' line should be changed to match section 6.2.4.1, p. 6-7 of the SunINGRES/Forms - VIFRED manual:

```
cc -c -w -I ingres/files filename
```

Work aroundt:
Use the correct 'cc' line which can be found on p. 6-7 of the SunINGRES/Forms - VIFRED manual.

Reference Number: 1005008
Synopsis: The Installation Manual and the RTF have incomplete /etc/rc entry
Release: SunINGRES 5.0

Description:
The Installation Manual and the RTF which contains some of the corrections for the /etc/rc entry for the lock daemon are incomplete. The complete entry should read:

```
# start the SunINGRES lock manager.
ING_HOME=/usr/prod/other/50/ingres; export ING_HOME
/usr/prod/other/50/ingres/bin/lockmgr    2> /dev/console
echo "SunINGRES lock manager started"
```

Reference Number: 1005014
Synopsis: 'Installing the lock daemon' Section omits file 'use.daem'
Release: SunINGRES 5.0

Description:

The section titled 'Installing the SunINGRES lock daemon' (section 3.4) of the 'SunINGRES Installation Guide' should mention that the file 'use.daemon' must be present in the directory ingres/files if one is to use the the lock daemon instead of the lock driver or instead of using 'single user' mode.

Library

Reference Number: 1004061
Synopsis: equel programs that have sunview can core dump under 3.2OS
Release: SunINGRES 3.0/25

Description:

An equel program that has sunview code was working under ingres3.0 and Sun OS 3.0. When recompiled under Sun OS 3.2 it core dumps while in MEfree during window_main_loop

General

Reference Number: 1000984
Synopsis: undocumented maximum length of path names for files referenced
Release: SunINGRES 2.0/23se19

Description:

If you try to compile an application that references files with a path name longer than 42 characters, you get the following ABF error:

```
9043: Error with parameters "file name".
```

Reference Number: 1000991
Synopsis: problem with using curses in an equel c program (Ingres)
Release: SunINGRES 2.0/23se27

Description:

Attempting to use Sun's curses routines in an equel C program will result in multiply defined statements.

Reference Number: 1000993
Synopsis: Equel program name length
Release: SunINGRES 2.0/23se27

Description:
Equel filenames having more than 14 characters will not work with the EQC preprocessor. The following error message results:
Illegal system filename "testtesttest.qc" used.

Reference Number: 1000994
Synopsis: Bug in doing Joindef's, QBF, over Network/Ingres
Release: SunINGRES 3.2, 3.0beta, 3.0

Description:
Doing Joindef's over Ingres Net causes SunINGRES to hang.
Work around:
rlogin into the backend machine and run QBF.

Reference Number: 1003890
Synopsis: f4 fields in a JoinDef can cause writes to hang
Release: SunINGRES 3.0/25

Description:
Using a field of type f4 in a JoinDef where the display format is altered causes SunINGRES to hang when you attempt to append a record to the database
Work around:
define the field as an f8 if you need to alter the display format

Reference Number: 1003891
Synopsis: A join field in a JoinDef won't get modified on a detail row
Release: SunINGRES 3.0/25

Description:
Given a JoinDef in which the joinfields are allowed to be modifiable for both the master and the detail, when the user modifies this field and attempts to write it back to the database, the value will only be changed for the master and not for the detail row.
The detail row is changed only if there was a modification made that row.
Work around:
Set up a dummy field in the detail to be checked if you want the change to carry over to the detail rows.

Reference Number: 1005507
Synopsis: Ingres 5.0 installation script fails for upgrades.
Release: SunINGRES 5.0

Description:
The installation script that is used to upgrade from SunINGRES release 3.0 to release 5.0 has the following problems:

1) Old binaries that are no longer needed are not deleted.
2) The library 'frunlib' is not deleted.
3) The files ingres/bin/kill_ing and ingres/bin/ntproc are not upgraded since they are owned by 'root', and the user 'ingres' does the install.
4) The files ingres/lib/* are not upgraded either because the permissions are 444.

Because the libraries are not upgraded, 'ranlib' generates error messages.

Work around:
Remove ingres/bin and ingres/lib before the installation.

Program

Reference Number: 1002664
Synopsis: optimizedb core dumps on a table with a text field
Release: SunINGRES 3.0/25

Description:
If 'optimizedb' is run on a table that contains a text field, SunINGRES core dumps.

Sun Common Lisp

Sun Common Lisp

Reference Number: 1004094
Synopsis: crash on error when using window system within editor
Release: Lisp2.0

Description:

When the Lisp window system is used from within the editor, Lisp aborts (to shell level) when attempting to handle certain user errors related to the window system. When the Lisp window system is used outside the editor, however, these errors seem to be handled properly.

Modula2

Reference Number: 1003608
Synopsis: Modula: referencing procedure as array crashes compiler
Release: modula1.0

Description:
Using brackets instead of parentheses in a procedure call causes the Modula-2 compiler to loop and use up disk space. The compiler crashes once it runs out of disk space on root.
Work around:
Find and fix the syntax error.

Reference Number: 1003943
Synopsis: -f68881 and -O cause assembler error in modula2
Release: modula1.0

Description:
Compiling the following with -f68881 and -O options produces an illegal instruction in the assembler code:

```
MODULE opt;
VAR r1, r2: REAL;
BEGIN
      r1 := ABS (r2);
END opt.
```

Reference Number: 1005347
Synopsis: A reset at the beginning of a file causes an access error
Release: modula1.0

Description:
Using 'reset' at the beginning of a file that is opened 'read-only' causes an access error.
Work around:
Open the file with 'read-write' access.

PC-NFS

PC-NFS

Reference Number: 1001424
Synopsis: incompatability problem with user software (xtree)
Release: pc-nfs1.0

Description:
Site specific information: PC type PC-AT
Hardware Configuration: Main memory size: 2M
Software Configuration:
Resident Software: pc-nfs software
Application program being used when problem occured: xtree
Network environment:
Tranceiver type: 3-com
Thick Ethernet
File server: Sun Microsystem 3/180
Number of PCs being supported by server: 2

xtree, a utility by Executive Systems does not work correctly on pcnfs mounted file systems. This is a program that graphically shows the directory structure and files on the disk. One of the options is to view the file. It works fine on the local hard disk or on a floppy, but does not work at all on the nfs-mounted file system.

Transcript

Transcript

Reference Number: 1003229
Synopsis: lpr fails to print full last character of long lines
Release: transcript2.0

Description:
Printing long lines of text using 'lpr' with Transcript Release 2.0 causes the rightmost character (the 83rd character on the line) to be partially printed, thus losing the rightmost portion.
Work around:
Use 'enscript' instead of 'lpr', if possible.

Reference Number: 1003644
Synopsis: daemon filter 'f' malfunction (2) error
Release: transcript2.0

Description:
When TranScript filters become confused by incorrect PostScript files, thus logging a message similar to the following:

```
Daemon Filter 'f' Malfunction (2)
```

Once the error message appears, printing becomes erratic.

SunSimplify
Library

Reference Number: 1004443
Synopsis: query_modify returns incorrect count
Release: SunSimplify1.0

Description:
The Eric procedure 'query_modify' incorrectly returns the number of records selected rather than the number of records that were updated because it parses the record counts incorrectly. For example, the following query returns a count of '2' instead of '0':

```
sql> delete emp where elongname = 'a*'/
recognized update!
2 record(s) selected, 0 record(s) deleted
```

Reference Number: 1004570
Synopsis: RPC timeouts possible for large databases
Release: SunSimplify1.0

Description:
It is possible to get 'RPC timeouts' when only a few users are accessing a large database located on one database server.

Program

Reference Number: 1003975
Synopsis: schemaload can core dump with delete_tables option
Release: SunSimplify1.0

Description:
The following code appears in remove_tables:

```
/* search thru field table for reference field */
if (seq_open(record, DB_FIRST, &scan_id) != DB_SUCCESS) {
        return(errno = DBERROR);
}

while (seq_next(&scan_id) == DB_SUCCESS) {
  .....
```

When performing schemaload, passing the address of scan_id to seq_next caused schemaload to seg fault when delete_tables is specified.

Reference Number: 1004123
Synopsis: dbload does not handle floats larger than 179
Release: SunSimplify1.0

Description:
dbload cannot handle floating point field specified with lengths longer than 179. This value is intended to be for screen display purposes only; larger float values should be saved. When attempting a SunUnify dbload, the following error message appears:

```
db can not convert - field value can not be created.
```

Reference Number: 1004337
Synopsis: databrowse core dumps when entity record locked
Release: SunSimplify1.0

Description:
When the user sets 'Skipped Locked Records' to 'yes' and then attempts to display a locked entity and select it, Databrowse core dumps.
Work around:
Do not run with 'Skipped Locked Records' set to 'yes'.

SunUNIFY

SunUNIFY

Reference Number: 1002645
Synopsis: btree index are getting corrupted
Release: SunUNIFY1.0

Description:
In Release 1.0 of SunUNIFY, B-trees can be corrupted if there are many users updating (or in a few cases reading) the B-trees concurrently.

Reference Number: 1003907
Synopsis: it is not possible to close sets opened by unisort
Release: SunUNIFY2.0

Description:
You cannot close a set opened via unisort. There is a limit of 8 sets that can be opened. You get an error after 8. The only way to close the sets is to exit the process.
Work around:
Exit process to clear sets.

Reference Number: 1005407
Synopsis: inconsistencies during rec_insertkey on deleted, locked record
Release: SunUNIFY2.0

Description:
Inconsistencies sometimes occur when two processes have the same record locked. Process A deletes the record, process B tries to do a 'rec_insertkey', but it fails because the record is locked. When the record is unlocked, the 'rec_insertkey' (your addrec) will work. The error returned when the record is locked is a bad reference; it is not an indication that the record is locked.

Reference Number: 1003000
Synopsis: SQL queries that use 'lines 0' and 'group by' cause wrong results
Release: SunUNIFY1.0, SunUNIFY2.0beta

Description:

If you use 'lines 0' at the start of an SQL query that has a 'group by' clause, the query appears to return incorrect results. For example if you do the following query:

```
select Manufacturer_ID
from manf
where State =
select State
from manf
group by State
having count(*) > 1/
```

You get these results:

```
Manufacturer_ID
----------------
        100
        101
        103
```

If you start out the same query with:

```
lines 0
```

You get the following result:

```
105
```

Reference Number: 1004479
Synopsis: SQL 'order by' gives incorrect results in special cases
Release: SunUNIFY2.0

Description:

When an 'order by' clause contains a column that is not in the target list and it is used with 'lines 0' at the start of an SQL query, SQL gives incorrect results.

Reference Number: 1003026
Synopsis: backend process not recognizing CTRL-Z via nfs mounts - dbon problem
Release: SunUNIFY1.0

Description:
If on(1C) is invoked by running Unify's SQL (it does a dbon),
then entering a Control-Z will hang the window over nfs mounts.

Reference Number: 1004287
Synopsis: 'afa' on a reference field causes invalid key msgs. on data entry
Release: SunUNIFY2.0, SunUNIFY1.0

Description:
You have a table that contains reference fields. These fields have
a default value placed on them in the referencing table. When
you try to add the key field, Unify will not accept this value and
gives an error message: Invalid key. There is no 'afa' entry
for the key field.
Work around:
Remove the 'afa' entries

Reference Number: 1004301
Synopsis: If a field contains a [a search using an exact match will fail
Release: SunUNIFY2.0, SunUNIFY1.0

Description:
You have a field in UNIFY which is a string. If you input data
with a left bracket ([), UNIFY is unable to find this when doing
a search on an exact match in either ENTER or on SQL. If
in ENTER you have a field that is 'this is a [test]' and a
field that has 'this is a test' and then you try to execute
an exact query on the first one, the second one
is what is found. In SQL no matches are found.
Work around:
Use either * or ? where the [should go.

Reference Number: 1001601
Synopsis: 'uidxrch' errors during SQL

Description:
SunUNIFY supports only one B-tree search at a time on a given
B-tree index by a given process. SQL violates this rule for
certain queries where a table is joined to itself. This
causes SQL to exit suddenly with the following error:

There was an error in opening the index file.
An internal system error has occurred.
Please check the error log.

The error log indicates an error in 'uidxsrch', which is
actually caused by 'opnbts' returning BTAOP_R
(B-tree index already open).

The problem occurs for queries like:

```
select <whatever> from my_table t1, my_table t2
    where t1.btfld = <whatever> and t2.btfld = <whatever>/
```

where 'btfld' has a B-tree index defined on it and SQL chooses
to access both t1 and t2 by that index.

Work around:

Workaround is to reformulate query so SQL does not choose to use
B-tree for more than one instance of table. Expressing self-join
as nested select works all the time; other formulations may or
may not work depending on what path the SQL optimizer chooses.

Reference Number: 1001617
Synopsis: sfmaint only displaying one page of information
Release: SunUNIFY1.0

Description:

When trying to display screen form coordinates (sfmaint) only one
screen of information is displayed even though more information
exists.

Work around:

Workaround is to check the screen field definitions using modify
(instead of inquire) mode; Next Page works in that mode.

Reference Number: 1002561
Synopsis: SQL does not handle (complicated) queries
Release: SunUNIFY1.0

Description:

Occasionally, posing a complex query for which the result depends on
the order of the second and third clause gives incorrect
results. For example, in the following complex query,
the third conjunct is ignored:

```
select bugid_value, bugid_keywd, keyword_val
from bug_tbl, kywd_tbl
where
    manager_value = 'pfeiffer'
  and
    bugid_keywd = bugid_value
  and
    [ keyword_val = 'setup' or keyword_val = 'install' ]
/
```

Reference Number: 1002644
Synopsis: ^C during a query that is on a btree index will cause problems
Release: SunUNIFY1.0

Description:
If a customer tries to run a query on a field that has a btree index built on it and then interrupts with a ^C, he is unable to run another query on that record until he backs out of that menu and starts it up again.

Reference Number: 1002986
Synopsis: Time fields on right side of nested sql return wrong info
Release: SunUNIFY1.0

Description:
Using a field of type time on the right side of a nested query will cause no records to be found even though some should have been.

Reference Number: 1003165
Synopsis: sql where clauses fail on 'const' = field
Release: SunUNIFY1.0

Description:
Consider the following typescript session, and note how the order is critical. For correct results one must query or field = const, and not const = field:

```
Sun UNIFY SQL -- VERSION 1.0
Copyright Unify Corporation 1983,1984,1985
Copyright Sun Microsystems, Inc.  1986
```

```
sql> select * from ccl_t/
recognized query!

concl_value|conclusion_desc
will fix    | bug is open
not a bug   | bug is closed
      .
      .
      .
from customer  |the bug is from customer
sql> select * from ccl_t where 'will fix' = concl_val/
recognized query!
There were no records selected.
sql> select * from ccl_t where concl_val = 'will fix'/
recognized query!

concl_val|concl_desc
will fix    | bug is open
sql>
```

Work around:

Change the order of the equality test.

Reference Number: 1003230
Synopsis: Help screen cause reference fields to get corrupted on screen
Release: SunUNIFY1.0

Description:

You have an ENTER screen for a table that contains fields which are references to another table. You then call up a help screen that is drawn over these fields. When the help screen goes away, these fields are not correctly refreshed.

Work around:

Arrange the layout of your screen and help screens to avoid this problem.

Reference Number: 1003872
Synopsis: sql query might ignore (trailing) AND clause
Release: SunUNIFY1.0

Description:

Consider the following query:

```
select count(*)
from bg_cal_t
where
        comp_call = 'other'
    and
        bugid_call =    select bugid_other
                        from other_t
                        where field_numb_other = 255
                        ;
/
```

There is no field_numb_other with value 255, hence the second AND clause is not satisfied. However, when the query is run in the above form, it acts as if the second AND clause is missing. Simply swap the order of the AND clauses and the query returns the correct result, 0.

Work around:

Stating the more complicated clause first helps.

Reference Number: 1003931
Synopsis: SQL finds matches where there are none
Release: SunUNIFY1.0

Description:

Consider the following two queries:

```
1.)  lines 0
     select 'bad submit', b_val
     from b_tbl
     where prog_val = 'submitted'
      and b_val in
          select b_pi
          from pi_tbl
          where sub_date = **/**/**
          ;
     /
```

There are no pi_tbl entries with null submit dates; therefore, there can be no records selected. In fact, if you run just the second select, sql is smart enough to state that no records were selected. Yet the full query will yield "satisfying" ids, which is false.

2.)
```
    lines 0
    select 'bad checked', b_val
    from b_tbl
    where
        b_val in
            select b_pi
            from pi_tbl
            where checked_date = **/**/**
            ;    and
        [ prog_val = 'checked' ]
    /
```
There are no b_tbl entries with prog_val set to 'checked'. Therefore, there can be no records selected. Yet this query produces results. There are many records which have a null checked_date.

Work around:

Rephrasing the query to the following seems to work:
```
    lines 0
    select 'bad submit', b_val
    from b_tbl
    where
        b_val in
            select b_pi
            from pi_tbl
            where sub_date = **/**/**
            ;
      and
        prog_val = 'submitted'
    /
```
In the second case, rephrasing the query to the following seems to solve the problem:
```
    lines 0
    select 'bad checked', b_val
    from b_tbl
    where
        [ prog_val = 'checked' ]
      and
        b_val in
            select b_pi
            from pi_tbl
            where checked_date = **/**/**
            ;
    /
```
Hence, it appears that putting the failing clause first is what is required. Note that the successful query is very context sensitive, making it difficult to pose queries that are good for varying cases.

Reference Number: 1004164
Synopsis: SQL loses editor buffer if CTRL-C is entered
Release: SunUNIFY1.0

Description:
SQL loses edit buffer if ^C (CTRL-C) is entered
Work around:
Save the editor buffer to a named file.
After the sql> prompt just enter
```
   start 'filename'
```

The problem may not be so apparent if sql is invoked within the Unify menu system.

Reference Number: 1004232
Synopsis: invalid field number from SCHENT (procfld) during schema entry
Release: SunUNIFY2.0

Description:
When the user changes the size of a newly added reference field, the referencing field's mode is set to 'M'. The mode should have remained 'A'.
Work around:
Delete the referencing field.

Reference Number: 1004259
Synopsis: SQL does string comparisons incorrectly
Release: SunUNIFY2.0

Description:
It is not possible to compare equal strings of unequal size in SQL and get a match when both strings are variables. They should match when both are left justified because of blank padding, but they don't. It is also possible to get strings to match incorrectly when both are variables and one has an asterisk at the end of it. The problem is present in 4.0 source code. In addition, unequal sized blank string constants are not equal, which is incorrect according to the standard (test with where ' ' = '   ').

The string 'abc' ^= 'abc   ', yet 'abc' = 'abc*' when 'abc*' is the value of a field and not a constant.
Work around:
Make all string fields that must be compared with another field (and not a constant) equal in size.

Reference Number: 1004764
Synopsis: ordering of SQL and clauses gives different results
Release: SunUNIFY2.0

Description:

The following SQL query returns incorrect values when the line marked with 3 stars is between the 'and category_val ^=' line and the 'and bugid_value ='. However, correct values are returned when the line marked with stars is the last line in the 'where' clause.

```
lines 0
select
        'Total Resolved', count (*)
from
        bug_tbl
where
  bugid_value > 999999
        and category_val ^= 'test'
        and bugid_value = select unique bugid_call
                    from bg_cal_t
                    where comp_call ^= 'Sun*';
***      and progress_val = < 'released', 'inactivated' >
/
select
        'Total Unresolved', count (*)
from
        bug_tbl
where
  bugid_value > 999999
        and category_val ^= 'test'
  and progress_val ^= 'released'
  and progress_val ^= 'inactivated'
        and bugid_value = select unique bugid_call
                    from bg_cal_t
                    where comp_call ^= 'Sun*'
/
```

8

CUMULATIVE INDEX: 1987

8

CUMULATIVE INDEX: 1987

Index

Revision History

Revision	*Date*	*Comments*
FINAL	August 1987	Seventh issue of Software Technical Bulletin (Software Information Services).

Bulk Rate
U.S. Postage
PAID
Permit No. 515
Mountain View, CA

Corporate Headquarters
Sun Microsystems, Inc.
2550 Garcia Avenue
Mountain View, CA 94043
415 960-1300
TLX 287815

For U.S. Sales Office locations, call:
800 821-4643
In CA: 800 821-4642

European Headquarters
Sun Microsystems Europe, Inc.
Sun House
31-41 Pembroke Broadway
Camberley
Surrey GU15 3XD
England
0276 62111
TLX 859017

Australia: 61-2-436-4699
Canada: 416 477-6745
France: (1) 46 30 23 24
Germany: (089) 95094-0
Japan: (03) 221-7021
The Netherlands: 02155 24888
UK: 0276 62111

Europe, Middle East, and Africa, call European Headquarters:
0276 62111

Elsewhere in the world, call Corporate Headquarters:
415 960-1300
Intercontinental Sales