4D1-3.3 Development Release and Installation Notes

IRIS-4D Series

SiliconGraphics
Computer Systems

Document number: 007-3357-010

4D1-3.3 Development Release and Installation Notes

Document Version 1.0

Document Number 007-3357-010

11/90

Technical Publications:
Chris Davidson
Lorrie Williams

Special thanks to the Silicon Graphics Engineering Staff.

4D1-3.3 Development Release
and Installation Notes
Document Version 1.0
Document Number 007-3357-010

Silicon Graphics, Inc.
Mountain View, California

Contents

List of Tables

1. Introduction

This document contains the procedure for installing your software, and supplemental information that was not available when the standard 4D1-3.3 documentation was published. In addition to this introduction, you will find information on these major topics:

- how to install your software
- major enhancements of this release
- compatibility and configuration information
- additions and changes to the software
- bugs that have been fixed
- known problems and how to work around them
- documentation errors and notes

These release and installation notes contain information that is vital to properly update your system to software Release 4D1-3.3. If you have a system that already is running this release of the software, Chapters 7 and 8 will be most useful to you.

Silicon Graphics, Inc., provides a comprehensive product support and maintenance program. For further information, please contact your local service organization.

2. Installing Software

This chapter explains how to install software on your IRIS-4D Series workstation or server. It covers:

- things to consider before you begin
- setting up the installation tools
- installing your software
- finishing up after you install new software
- troubleshooting installation problems

2.1 Read This BEFORE You Begin

This section contains important information and explains several things you should consider before you start to install software on your IRIS-4D Series workstation or server. Reading this section before you begin will help you avoid problems and confusion as you install your software.

To install new software, you must shut down your system (if it's running) and install special software installation tools. These tools include *inst* and a temporary operating system called the *miniroot*. Carefully read this section before you begin the software installation procedure, because it covers some topics that need to be considered *before* you shut down your system.

Your tape drive should be clean. Most tape read errors are caused by dirty drives.

Important Before you begin to install your new software, make a complete backup of your system. See your owner's guide for instructions on making backups.

Network Addresses

If you plan to install software from a remote tape drive or a directory on a remote system, *inst* might prompt you for the IP network addresses of your system and, possibly, the name of the remote system you use for installation. *inst* looks to your */etc/hosts* file for these addresses. However, if the file doesn't exist, or if you are referencing a remote system not noted in your system's */etc/hosts* file, *inst* prompts you for the IP network addresses of the systems in question.

Note: If you want to install software remotely over a network through a gateway system, the systems must be able to communicate across the network. Check the file */usr/etc/inetd.conf* on the gateway system to make sure that the last column of the *bootp* entry is:

```
bootp -f
```

See "The *distcp* Command" in these release notes, your Owner's Guide, and the *Network Communications Guide* for more information about using the network.

Once your system is shut down to the PROM level to install software, you do not readily have access to these addresses. Therefore, before you begin to install software, find and record the IP network addresses of the workstations or servers directly involved with your installation.

You can get these addresses by looking at any */etc/hosts* file on your network. You are looking for four numbers separated by periods. For example, a typical network address might look like this:

```
192.35.73.96
```

If you have a workstation named solaris, either of the following commands should provide you with what you need:

```
grep solaris /etc/hosts
```

or if your system is running Yellow Pages:

```
ypmatch solaris hosts
```

You should see something similar to this:

```
192.35.73.96     solaris.foo.com  solaris
```

Record the network addresses in case you need them.

Installing Earlier System Software Releases

Once you've installed the 4D1-3.3 system software, you cannot re-install a previous system software release (or portions of a previous release) without first completely removing IRIX 4D1-3.3. If, for some reason, you must re-install a previous system software release, make a complete backup of all user data, reboot your system with the distribution tape of the earlier release and use the *clean* option in *inst* from that earlier release to erase all 4D1-3.3 system and user files. You can then restart the installation procedure from scratch. When you have finished re-installing the previous release, use the backup tapes you made to put the user data back on the system. Read the following section for important information on the compatibility of installation tools across different system software releases.

Note: The on-line installation history format is slightly different in Release 4D1-3.3 than it was in previous releases. When you use the 4D1-3.3 installation tools, the on-line installation history will contain this new format. Once this is done, you cannot use older versions of the installation tools because they will not understand the new format. If it is necessary to install older software once your system has been updated to 4D1-3.3 or newer, use the new installation tools to do so.

Determining Which Installation Mode to Use

There are two modes for installing software: *automatic* and *manual.* The easiest way to install software is to use the *automatic* mode. The *automatic* mode will replace all out of date subsystems with the current corresponding subsystems. It will not install subsystems which are not already installed unless those subsystems are considered mandatory. If it fails (for lack of disk space, for example) or if you want more control over your installation than the *automatic* mode provides, you should use the *manual* mode.

Software is divided into groups of related files called *subsystems.* The *manual* installation mode lets you preview a list of the available subsystems, the subsystems selected for installation, and the ones that have not been selected at all. The *manual* mode also provides other functions that you might need during installation. Use the *manual* mode if you need to:

- use less disk space than is required for installing the currently selected set of subsystems
- override the default subsystem selections
- verify *inst*'s default subsystem selections
- specify (or respecify) the installation source. This is a place, such as a tape drive (device) or directory, in which components (such as software images and product descriptors) of a product reside. A product descriptor file is a list of the components in a product that *inst* uses to install that product.
- make new file systems, control file system mounting or unmounting, or perform additional network initialization tasks
- delete unnecessary subsystems

Special Characters Used in Installation

The software products you receive from Silicon Graphics, Inc., are divided into discrete components, which you must deal with before, during, and after installation.

A broad, internal division of a product, such as on-line manual pages, is called an *image.* An image is divided into *subsystems,* which are made up of related files. Each subsystem has a name that follows this format:

```
    product.image.subsystem
```

For example, *ftn.man.fedgetut* is the name of the Fortran *edge* on-line manual page subsystem.

Using the *manual* mode, you can specify on which subsystems you wish to operate. Often, you will need to specify more than one subsystem at once.

You can specify a subsystem name individually or use a ''pattern'' to specify many at one time. A pattern is a subsystem name that contains shell-style metacharacters. These metacharacters allow you to match zero or more regular characters in the available subsystem name(s).

The metacharacters are:

?	matches one character
*	matches any combination of characters
[ ]	matches any enclosed characters or range of characters separated by a dash

Missing components in subsystem names are treated by the system as though asterisks (*) are in their places. The following list contains two hypothetical products, to help you see how special characters and missing components affect subsystem selection. Both products, *abc* and *xyx,* contain two images: one for manual pages (on–line documentation) and one for software.

In this example, each image is divided into five subsystems:

```
abc.man.prgs      xyx.man.prgs
abc.man.games     xyx.man.games
abc.man.demos     xyx.man.demos
abc.man.tools1    xyx.man.tools1
abc.man.tools2    xyx.man.tools2
abc.sw.prgs       xyx.sw.prgs
abc.sw.games      xyx.sw.games
abc.sw.demos      xyx.sw.demos
abc.sw.tools1     xyx.sw.tools1
abc.sw.tools2     xyx.sw.tools2
```

Here are examples of some names and patterns, and the subsystems they identify:

Name	Identified Subsystems
abc	all subsystems in the *abc* product
abc.man	all subsystems in the manual page image of *abc*
**.man*	all subsystems in all manual page images
*.**.demos*	all demo subsystems in all products
.sw.[0-9]	all subsystems in the software images of all products that end with a digit

See Section 2.4.2, ''The Subsystem Selection Menu,'' for more information on selecting subsystems.

2.2 The Installation Procedure

This section describes how to prepare your system for installation and use *inst* to install new software.

2.2.1 Setting Up the Installation Tools

To install software on an IRIS-4D Series workstation or server, you first need to set up the installation tools. The main component of this tool set is the *miniroot*. The *miniroot* is a small, temporary operating system that prevents *inst* from overwriting system files that would be active if the IRIX operating system were running.

When you start setting up the installation tools, you will have to bring the system down to the PROM level. Once the *miniroot* is set up, it automatically invokes *inst*, at which point you can install new software. When you've finished installing your software and reboot your system, IRIX overwrites the *miniroot* (which is stored in *swap*), and your newly installed software is ready for use.

In the event of a power failure during this procedure, rebooting the system returns you to the *miniroot* and *inst* by default. To quit out of *inst* after a power failure, you may select the *quit* option from any of the installation menus and your original system configuration should be restored. If the system does not recover properly, see ''Error Recovery,'' later in this chapter.

Warning: Some systems include a bank of eight switches on the front panel for use in certain debugging operations. If you have such a system, make sure that these switches all point away from the word OPEN before you bring your system down to the PROM level. If you change the switch position, you must reset the system and then reboot.

1. If your IRIS-4D Series workstation or server is not already running, turn it on and bring it to PROM menu. See your owner's guide if you need help with this. If you see this message:

   ```
   Press any key to restart.
   ```

   Press any key on the keyboard. If you see a message similar to this one:

   ```
   Starting the system, press <Esc> to stop.
   ```

   Press **<Esc>**.

2. If your system is already running, log in as *root* and then use *who*(1) to determine whether anybody else is logged into the system. If so, post a system-wide message with *wall*(1) asking them to log off while you install new software. Give people a few minutes to finish what they are doing and to log off.

   If you have not already done so, make a complete backup of the system. See your owner's guide or the *IRIS-4D System Administrator's Guide* for more information about making backups.

   Type this command to shut down the system:

   ```
   /etc/halt
   ```

3. If your system displays this:

```
     Starting up the system...
To perform system maintenance instead, press <Esc>.
```

   Press **<Esc>**.

   Depending on the type of IRIS-4D Series workstation or server you have, you see either the PROM Monitor prompt, which looks like this:

```
>>
```

   or the System Maintenance Menu.
   These conditions require that you begin you installation at Section 2.2.3, "Using the PROM Monitor":

   - you see the PROM monitor prompt >>
   - your system has more than one tape drive (you need to specify which tape drive to use for installation)

   If none of these conditions are true, continue on to step 4.

4. If you plan to use the tape drive on your system, select number 2, "Install System Software."

   If you plan to install software over a network (i.e., from a remote system) and *your* system has a tape drive, select 5, "Enter the Command Monitor" and type the following at the Command Monitor prompt:

```
setenv notape 1
```

   Then, type **exit** to return to the System Maintenance Menu. Select number 2, "Install System Software," and go to step 5.

5. If you see this message:

```
Insert the installation tape.
```

Go to step 6.

If you see this message instead:

```
Are you using a remote tape? (y/n):
```

you may:

- type **n** if you plan to install software from your system's tape drive, in which case you see:

  ```
  Enter the name of the machine...
  ```

  Enter the name of your system and go to step number 6.

- Type **y** if you want to use the tape drive on a remote system. You see:

  ```
  Enter the name of the machine...
  ```

  Enter the name of the remote system and go to step number 8.

- Type **n** if you want to install software from a directory on a remote system. You see:

  ```
  Enter the name of the machine...
  ```

  Enter the name of the remote system and the distribution directory in this format:

  `solaris:`*/dir_path*

  Replace **`solaris`** with the name of the computer from which you plan to copy your software. Replace *dir_path* with the path name of the directory where the software image resides. Press **`<enter>`**.

6. You see a message similar to this:

   ```
   Insert the installation tape, then press <enter>:
   ```

   Insert the Execution Only Environment 1 tape (EOE1), lock it in place, and then press **`<enter>`**.

7. You should see this message followed by several lines of dots that print across your screen as the installation software is loaded onto the system:

```
Copying installation program to disk.
```

After several minutes, the dots stop printing across the screen, and the system boot process begins. At this point, you may see some messages similar to this:

```
xyl757 ctlr 3:missing

enp0:missing
```

These harmless messages may be ignored.

When the system boots, you see the *inst* menu, which indicates that *inst* is running and that you may install your software. You see:

```
Ready to install software.
 Choose an item, then press <enter>:
   1. Automatically install software
   2. Use manual installation features
   3. Help
   4. Quit
```

Go to Section 2.2.2, ''Using *inst*'' for further instructions.

Note: If the system has trouble reading the tape, see section 2.1, ''Network Addresses''.

8. Continue here if you are installing from a remote tape.

Before you type anything, make sure the installation tape is locked in the remote system's tape drive. On the remote system, type:

```
mt rewind
```

9. On your system, type the name of the remote system, then press **<enter>**. You see:

```
Copying installation program to disk...
```

After a few minutes, the dots stop printing across the screen and you see the *inst* menu, which indicates that *inst* is running and that you may install your software. You see:

```
Ready to install software.
 Choose an item, then press <enter>:
   1. Automatically install software
   2. Use manual installation features
   3. Help
   4. Quit
```

Go to section 2.2.2, ''Using *inst*'' for further instructions. If there is a problem at this point, see the section on ''Remote Installation Failure'' later in this chapter.

2.2.2 Using *inst*

Once *inst* is automatically invoked, installing software is quite straightforward. This section explains how to use *inst* to install new software on your IRIS-4D Series workstation or server.

1. *inst* displays the main installation menu on your screen.
2. If you are installing from a tape cartridge, insert the tape in the tape drive and lock it in place.
3. Choose installation mode from the menu. Remember, you can install most software automatically. This is the simplest way to install software.

   If *automatic* installation fails (because there is not enough space on the hard disk, for example), or you want interactive control over what you install on your hard disk, use the *manual* mode. See Section 2.1, ''Before You Begin'', to determine which installation mode you should use.

   Note: You can switch installation modes while you are using *inst*. This is useful when you want to install software from multiple sources.

   Enter the number, the name, or an abbreviation of the menu item you choose and then press `<enter>`.

   You can ask *inst* for *help* at any point during the installation process, except when software is actually being copied to the hard disk. Section 2.4, ''The Installation Menus'', provides detailed information on the *help*.

   inst runs a series of pre-installation checks to determine which subsystems to install by default. If you are installing a product for the very first time, the default subsystems are predefined. If you are updating a product, the default subsystems are those that replace previously installed, corresponding subsystems.

   New software directly replaces corresponding subsystems already on the disk though they might have different names from their older counterparts. *manual* mode allows you to redefine which subsystems are to be installed.

The pre-installation check tests to:

- ensure that mandatory subsystems are selected
- determine whether prerequisite subsystems are already installed
- determine whether enough space exists in the file system to install all of the selected subsystems

Note: If any of these pre-installation checks fail, you see a warning message and the installation stops. You can use the *manual* mode to resolve such problems and then continue installing your software.

If you are attempting to install from a remote tape or remote directory, *inst* might prompt you for network addresses. (See ''Network Addresses'' in Section 2.1 for help with this.)

4. If you selected the *manual* mode, skip ahead to step 7 to continue with the installation.

   If you chose to install your software automatically, the software is copied from the source to the hard disk. *inst* notifies you as old versions of subsystems are being removed and new versions are installed.

   If *inst* has trouble getting software across the network, refer to ''Network Addresses'' or ''Remote Installation Failure'' (whichever is applicable to your situation).

   At the end of the software installation, *inst* checks for compatibility between the new software being installed and existing software already on the system. If any incompatibilities are found, *inst* notifies you by printing out the names of the incompatible subsystems. These incompatibilities must be removed or replaced before *inst* will let you *quit*. For information on resolving incompatibilities, type:

   ```
   help incompatible
   ```

   When the installation process is complete, you see:

   ```
   Done.
   Is there more software to install?
   ```

5. Indicate whether to install additional software.

If you don't have any more software to install, type **no** and press **<enter>**.

If you have more software to install, type **yes** and then press **<enter>**. You see:

```
Insert the next tape, then press <enter>:
```

Insert the tape and press **<enter>**. You see the following menu again:

```
  Choose an item, then press <enter>:

       1. Automatically install software
       2. Use manual installation features

       3. Help
       4. Quit
```

Repeat the installation procedure until you are finished installing your software.

Note When you use *quit* to exit, *inst* checks for compatibility with existing software on the system. If any of the products are incompatible with the new software, *inst* notifies you of the incompatibility. Before you are allowed to quit *inst*, you must resolve the incompatibilities.

6. Restart the system.

   When all of your software is installed, you see:

```
Please wait...
Ready to restart the system.  Restart? [y, n]
```

Type `y`, then press `<enter>`. The system reboots, and your new software is available for use.

Next, you see a message reminding you to use *versions changed* to see which configuration files changed during installation. Go to Section 2.3, ''Finishing Up the Installation'', for further instructions.

7. Continue here if you are installing your software using the *manual* mode.

   The general procedure for using the *manual* mode is outlined below, but can vary depending on your particular needs. You must be familiar with the various menus to successfully perform an installation using the *manual* mode. See Section 2.4.1, ''Manual Installation Menus'' for a complete explanation of each menu item function.

8. Use *admin* to perform any necessary file system operations, such as making new file systems, mounting, and unmounting.

9. Use *admin* to perform any nonstandard network initialization that might be required.

10. Use *from* to specify or respecify the installation source if necessary.

11. Use *list* to examine a list of the subsystems to see which are available and which are selected by default.

12. Choose one of the *standard*, *all*, or *select* items. The *standard* and *all* items initiate installation. The *select* item invokes the Subsystem Selection menu. See Section 2.4.2, ''The Subsystem Selection Menu'', to learn more about how to use the *select* option.

13. Repeat steps 8 through 12 above for each additional software product that you are installing.

14. Choose *go*, if you used *select*) to copy the software to your disk. This instructs *inst* to install the selected software on your system.

15. When you are finished installing software, use *quit* to stop the installation tool. At the end of the software installation, *inst* checks for compatibility between the new software being installed and existing software already on the system. If any incompatibilities are found, *inst* notifies you by printing out the names of the incompatible subsystems. These incompatibilities must be resolved before *inst* will let you *quit*. For information on resolving incompatibilities, type:

```
help incompatible
```

16. Restart the system.

    When all of your software is installed, you see:

```
Please wait...
Ready to restart the system.  Restart? [y, n]
```

    Type `y`, then press `<enter>`. The system reboots, and your new software is available for use.

    You see a message reminding you to use *versions changed* to see which configuration files changed during installation.

Continue to Section 2.3, ''Finishing Up the Installation'', to modify any configuration files that were installed or moved during the installation.

2.2.3 Using the PROM Monitor

If your system displays the System Maintenance menu, choose number 5, ''Enter Command Monitor'' to get the PROM monitor prompt >>.

Follow the instructions that apply to your software installation in one of the following subsections:

- From a Local Tape Drive
- From a Remote Tape Drive
- From a Remote Distribution Directory

From a Local Tape Drive

To set up the *miniroot* from a local tape drive, follow these steps:

1. Put the installation tape in the tape drive.
2. Set the environment variable *tapedevice* to the name of your tape device by typing in the following command at the >> prompt and pressing **<enter>**. Replace *device_name* in the command below with the name of the device that corresponds to your tape drive. See Table 2-1 for standard tape drive device names.

   ```
   setenv tapedevice device_name
   ```

   You can use the *hinv* command to determine the SCSI number for your drive:

   ```
   hinv
   ```

   Table 2-1 lists the device names for QIC and SCSI tape drives. *n* is the controller number and *ID* is the ID number.

Type of Drive	Device Name
QIC tape drives	tpqic(*n*,*ID*)
SCSI tape drives	tpsc(*n*,*ID*)

   Table 2-1. Tape Drive Device Names

   Refer to the documentation from the tape drive manufacturer to determine which type of drive you have.

3. Boot the standalone shell, *sash*, from the tape device. Replace *cpu* in the command below with the appropriate CPU type from Table 2-2.

```
boot -f ${tapedevice}(sash.cpu) --m
```

Model Number	CPU Type
4D/60	R2300
4D/50, 60T, 70, 80, 85	IP4
4D/120	IP5
4D/20, 25	IP6
4D/220, 240, 280, 320, 340, 380	IP7
4D/210	IP9

Table 2-2. CPU Types

If you encounter errors in booting *sash*, see the section "Trouble Reading a Local Tape," later in this chapter.

This command copies the *miniroot* to your hard disk and boots your system from it. After about 10 minutes, you see this message:

```
Ready to install software.
  Choose an item, then press <enter>:
     1. Automatically install software
     2. Use manual installation features
     3. Help
     4. Quit
```

Now you can install your software. See Section 2.2.2, "Using *inst*", for further instructions.

From a Remote Tape Drive

A remote system must be running system software release 4D1-3.2 or later to load the installation tools from a remote tape drive. Follow these steps:

1. If the *netaddr* variable on your system is not set, or is incorrect, you need to set it. (Refer to "Network Addresses" in Section 2.1 for help doing this):

   ```
   setenv netaddr 192.35.73.96
   ```

2. Make sure the installation tape is locked securely in the tape drive on the remote system.

3. Set the *tapedevice* environment variable as shown in the example below. Replace **solaris** in the command line below with the name of the remote computer.

   ```
   setenv tapedevice bootp()solaris:/dev/nrtape
   ```

4. Boot the standalone shell, *sash*, from the tape on the remote computer. Replace *cpu* with the appropriate CPU type from Table 2-2 in the previous section.

   ```
   boot -f ${tapedevice}(sash.cpu) --m
   ```

   You see messages similar to these:

   ```
   Obtaining /dev/tape from server remote.
   Copying installation program to disk
   ```

   followed by a series of dots.

   If you have problems booting *sash*, see "Remote Installation Failure," later in this chapter.

After about ten minutes, you see this message:

```
Ready to install software.
Choose an item, then press <enter>:
     1. Automatically install software
     2. Use manual installation features
     3. Help
     4. Quit
```

Now you can install your software. See Section 2.2.2, ''Using *inst*'', for further instructions.

From a Remote Distribution Directory

To load the *miniroot* from a directory located on a remote computer, follow these steps:

1. Use *distcp* to prepare a directory on the remote computer from which you plan to install your software. Ths system containing the distribution directory must be running 4D1-3.3 in order to read the data from the 4D1-3.3 distribution tapes. See ''The *distcp* Command'' in Section 2.5 for more details.
2. Set the *netaddr* variable on your system. (Refer to ''Network Addresses'' in Section 2.1 for help doing this). To check the *netaddr* variable, type:

   ```
   printenv netaddr
   ```

   To set the *netaddr* variable, enter:

   ```
   setenv netaddr 192.35.73.96
   ```

3. Set the *notape* environment variable as shown below:

   ```
   setenv notape 1
   ```

4. Set the *tapedevice* environment variables as shown below. Replace *dir_path* with the complete path of the directory from which you want to install software:

   ```
   setenv tapedevice bootp()solaris:/dir_path/sa
   ```

5. Boot the standalone shell, *sash*, from the tape on the remote computer. Replace *cpu* with the appropriate CPU type from Table 2-2.

   ```
   boot -f ${tapedevice}(sash.cpu) --m
   ```

   You see messages similar to these:

   ```
   Obtaining /dir_path/sa from server remote.
   Copying installation program to disk
   ```

   followed by a series of dots.

   If there are problems at this point, see "Remote Installation Failure," later in this chapter.

   After about 10 minutes, you see this message:

   ```
   Ready to install software.
     Choose an item, then press <enter>:
        1. Automatically install software
        2. Use manual installation features
        3. Help
        4. Quit
   ```

   Go to Section 2.2.2, "Using *inst*" for further instructions.

2.3 Finishing Up the Installation

Adding and updating software often affects configuration files. The type and number of configuration files affected depend upon the product that you are installing.

When you finish installing software on your system, you might see a message reminding you to update your configuration files and remove obsolete ones.

The following list describes how *inst* handles configuration files that have been modified by the user prior to installation:

- If a configuration file on your system should not be updated and cannot be improved upon, *inst* changes nothing.

- If the product you are installing contains a new configuration file with optional improvements, *inst* leaves your files alone and installs the new file for comparison, with *.N* appended to the file name.
- If the product you are installing contains a new configuration file with mandatory improvements, *inst* installs the new file and renames your old one for comparison, with *.O* appended to the file name.
- Obsolete configuration files are renamed with a *.X* suffix. These files are *not* in the installation history of the product, but can be located with the following command:

  ```
  versions user | grep ".*\.X$"
  ```

  See "The *versions* Command", in Section 2.5 for more information.

You can find detailed explanations of the purpose and format of configuration files in the following documents:

- *IRIX System Administrator's Guide*
- *System Tuning and Configuration Guide*
- *Network Communications Guide*

Error Recovery

If *inst* terminates abnormally, the system and the on-line installation history file is abandoned in an undefined state. This renders the product being installed or removed unusable. In such cases, you should correct the cause of the error if possible, and then either re-install or remove the software product that was being installed (or removed) when the problem occurred.

If the system is reset or has a power failure during installation, the system automatically reboots into the *inst* tool. This reduces the chances of having a partially updated system, which would probably result in strange behavior or system crashes later.

In the event that this happens, all that is necessary to restore the old boot partition information is to *quit* normally from *inst*. If, instead, you attempt to reload the *miniroot*, you will get a warning from *sash* that the root and swap partitions are the same. This is usually due to an interrupted installation.

If all else fails, the boot partition must be reset either by *fx*(1) (standalone or kernel) or by *dvhtool*(1M) under UNIX. The *System Tuning and Configuration Guide* contains information on booting from an alternate root partition.

Trouble Reading a Local Tape

If there is a problem in the early stages of the installation process, while the PROMS are attempting to load the *sash* program or copy the *miniroot* to the disk, it might be th result of a problem in reading the tape. Retry the operation at least one more time. Also, try cleaning the tape drive.

If there is still a problem, and there is another system with a tape drive available, try to read the tape on the other drive:

```
mt rewind
dd if=/dev/nrtape of=/dev/null bs=16k
mt rewind
```

This serves only to read through the data of the first file on the tape and copy it ''nowhere'', and may take several minutes. You should see no error messages, other than a count of records copied in and out.

If the problem occurs later in the process, after *inst* is running and an attempt is made to begin installing software, you might want to verify that the subsequent files on the tape can be read. Note that access is always achieved through the no-rewind variant of the tape device. If you are specifying a non-standard tape device, make sure you are using the no-rewind device. Escape to a shell with the *sh* command, and issue these commands:

```
mt rewind
mt fsf 2
dd if=/dev/nrtape of=/dev/null bs=16k
dd if=/dev/nrtape of=/dev/null bs=16k
dd if=/dev/nrtape of=/dev/null bs=16k
dd if=/dev/nrtape of=/dev/null bs=16k
mt rewind
```

Each of the *dd* commands serves to read through the data of the next physical file on the tape. You should see no error messages other than a count of records in and out.

If the files cannot be read, there might be a problem with your copy of the installation tape and/or the tape drive. The most probable cause of this is a dirty tape drive.

If the files can be read, the tape and tape drive are probably OK. Exit the shell by typing *exit*. Issue a `from tape` command to cause *inst* to try and read from the tape again.

If the problem cannot be isolated using these techniques, call the Geometry Hotline.

Remote Installation Failure

If there is trouble reading from a remote tape drive early in the process, make sure that the tape is readable on the remote system's tape drive, as described in the section on ''Trouble Reading a Local Tape''. If the tape can be read properly, the problem is probably due to a poor network connection.

If you are attempting to install from a directory on a remote system, double check the installation source system name and path names and then assume that the problem may be due to the network connection.

The PROMS require that the *netaddr* variable contain your system's IP address. If the remote system is being accessed through a network gateway system, the PROMS also require that the *gateaddr* variable contain the gateway system's IP address. The techniques in "Network Addresses," will help you get these addresses. Use the PROM monitor prompt (>>) to examine this (with *printenv*) and/or change the current environment variables (with *setenv*). Double check the addresses and correct them as necessary.

If you are installing through a network gateway system, make sure the gateway system is configured to forward *bootp* requests. Refer to "Network Addresses", in Section 2.1.

If network access fails later in the process (for example, when *inst* is attempting to read the installation software), it might be due to a permissions problem. If you haven't explicitly specified an account name to be used on the remote system (through the *from* menu item), *inst* tries the *guest* account. If *guest* account access is not granted, this attempt fails.

When *inst* is running in the *miniroot*, it initializes TCP/IP just before the first attempt is made to read from the directory on the remote system. Once this occurs, you should be able to test various network connections using some of the standard TCP/IP commands. If you want to verify that a connection can be made to a particular system, through a particular user account, type `sh` at any *inst* prompt to escape to a shell and then issue this command:

```
rsh solaris -l user date
```

The remote system should respond with the current date. If this command fails, *inst* will also fail for the same reason. It might be that your system is not configured correctly. Examine the */etc/hosts* file.

If the error is permissions related, the user *inst* on your system is not allowed to use the *user* account on the remote system. There might be a more appropriate *user* account on the remote system. Sometimes, administrators set up a user account (and restricted shell) for use only by *inst*.

To direct *inst* to get software through a particular user account, use the *from* menu item:

```
from user@server:/dev/nrtape
```

or

`from` *user*`@`*server*`:`*dir_path*

If you wish, you can issue this command as soon as *inst* starts running. See Section 2.4.1, ''Manual Installation Menus,'' for more information on the *from* command.

Installing the Kernel Debugger

The following three error messages occur when a user has chosen to install *dev.sw.debug*, either in this, or an earlier installation. Part of the *dev.sw.debug* subsystem is the symbolic monitor portion of the kernel debugger *symmon.*

```
no room in volume header for symmon
[Aux cmd] if test @$instmode = @normal ; then /etc/dvhtool -v\
creat $rbase/stand/symmon symmon $vhdev ; fi

no room in volume header for sash
[Aux cmd] if test @$instmode = @normal ; then /etc/dvhtool -v\
creat $rbase/stand/sash sash $vhdev ; fi

no room in volume header for ide
[Aux cmd] if test @$instmode = @normal ; then /etc/dvhtool -v\
creat $rbase/stand/ide ide $vhdev ; fi
```

The first error occurs only while installing *dev.sw.debug*. If you do not plan to use the kernel debugger, you should simply not install this subsystem (or you may simply enter 'continue' on the interrupt menu, which completes the *dev.sw.debug* installation, but does not install *symmon* in the volume header).

If you need to use the symbolic debugger, you need to re-partition your drive with the 4D1-3.3 version of *fx*, and completely reinstall your system. All data on the disk is lost in this case, so you must back up your system first. See the *IRIX System Administrator's Guide* for information on how to repartition your disk.

The second two forms of the error message may occur during the installation of *eoe1.sw.unix*, if you have installed *dev.sw.debug* during an earlier installation.

These files are installed in both */stand*, and in the volume header of the disk. If the system's disk layout was configured before the 4D1-3.2 release, then the volume header may not be large enough to contain *symmon* in addition to *ide* and *sash.*

symmon was first shipped in 4D1-3.2, and at that time the disk partitioning tool was changed to create a larger default volume header partition.

In some cases, earlier volume headers may have been just barely large enough, but increased sizes of *sash* and *symmon* may cause them not to fit.

If the error occurs on *sash* or *ide*, you should also choose the your previously installed release still installed. If you need the new *sash* and *symmon*, you need to re-partition your disk (see above).

Otherwise, when the installation is complete, before quitting from the installation tool, use the *shroot* command, and then execute the command

```
dvhtool -v d symmon -v c /stand/sash sash
```

which removes *symmon* from the volume header, and replace the older *sash* with the new version (where 'sash' would be replaced in both places with 'ide' if the *ide* installation had failed).

Recovering from an Unbootable Kernel

For help recovering from an unbootable kernel, refer to Chapter 1 of the System Tuning and Configuration Guide.

2.4 The Installation Menus

inst provides a number of menus that let you control the installation process. Most of these menus apply only to using the manual installation features, but some, such as the Interrupt/Error menu, apply to both installation modes.

The menu items *help*, *set*, *quit*, *sh*, and *shroot* are available from every menu even though they are not listed on the menu. (Two exceptions to this are the *sh* and *shroot* items, which are never available from the Interrupt Menu.) These ''hidden'' menu items provide administrative support without cluttering up your screen. These items also are available from the the Administration Functions menu, which is covered in detail in Section 2.4.3.

Some menu items can be used only from the *miniroot*. For example, you can invoke items that affect file systems and network initialization only from the *miniroot*.

You can choose an item on a menu by name or by number, confirming the selection by pressing `<enter>`. The names of the menu items are the first words on the line and can be abbreviated if you like.

Some of the menu items accept an argument (a name or number), which affects the way the menu item reacts. *inst* prompts you for an argument if it expects one and you have not supplied it.

2.4.1 Manual Installation Menus

The Manual Installation menu contains these items:

from [source]

Specifies the distribution source. The three kinds of source are the no-rewind tape device, a specific product descriptor file in a distribution directory, or a distribution directory. Any of these can be local or remote.

If the distribution source is tape, *inst* tries to position the tape and read the *product descriptor* file. The product descriptor is an internal representation of the product, image, and subsystem hierarchy, providing all necessary information on the product to *inst*.

To prepare a distribution directory, use the *distcp* command to copy software products from tape to disk. You can use the product descriptor files in the directory as distribution sources. You can name the distribution directory itself as a source. If you do this, *inst* reads all product descriptors in the directory. For more information on preparing a distribution directory, see Section 2.5, "Special Features".

If the distribution source is on a remote system that you can access over an Ethernet via TCP/IP, prefix the source name with *host:* or *user@host:*.

Here are some example distribution source names:

```
Manual> from /dev/nrtape
Manual> from dserver:/dev/nrtape
Manual> from guest@dserver:/u/dir_path
```

If you encounter problems accessing the distribution, see ''Trouble Reading a Local Tape'' or ''Remote Installation Failure'' in this chapter.

list [names]

Lists information for the subsystems of the current distribution source. By default, all known subsystems are listed. You can use names or patterns to restrict the listing to specific subsystems.

There are also several special keywords that you can use to identify the subsystems you want to list, including all of the keywords in Table 2-3 (in ''The Subsystem Selection Menu'' section) and the following:

Keyword	Function
products	lists products, not subsystems
images	lists images, not subsystems
sizes	list absolute subsystem sizes (rather than disk space deltas

All remaining items are taken as subsystem names or patterns.

Each line in the output of the list menu selection describes one subsystem and contains several columns.

If the subsystem is currently selected for installation, the first column is ''i''. If a version of the subsystem is already installed and will be left alone during this installation, the first column is ''k''. If an ''r'' is present, then the subsystem will be removed from your machine.

If the subsystem is already installed on the disk, the second column is ''I''. If a subsystem from a previous release has been installed on the disk, the second column is ''X.'' An ''N'' in this column indicates that you are trying to install an older version of the subsystem than what is currently installed on your machine. If there is no related version on the disk, the second column is blank.

	The third column is the subsystem name in the form *product.image.subsystem*. This name identifies the subsystem to the various menu items during subsystem selection.
	The next columns are the disk space "deltas" for the subsystem. There is one column for the parent directory of each file system, usually / and */usr*. The heading shows the parent directory names. The delta is a number followed by a plus or minus sign to indicate the estimated change in disk space, in 512 byte blocks, that would result from installing that subsystem. (The numbers are estimates based on size of the existing files on the disk, and the size of the new files to be installed.) These numbers can be negative, indicating that the new subsystem requires less room on that file system than the subsystem(s) it replaces.
	The final column is the description of the subsystem, indicating the function of the files in the subsystem.
	If the *verboselist* option is set, the subsystem list includes the product and image names, delta totals, and descriptions, along with the subsystems they contain.
standard	Installs the default set of subsystems. *inst* performs the standard pre-installation checks, then begins installation.
all	Installs all of the subsystems that are supplied with the distribution.
select	When both *standard* and *all* are inappropriate, use the *select* item to invoke the Subsystem Selection menu, from which you can choose the specific subsystems you want to install, and perform other related functions.
recalculate	Recalculates the disk space deltas. If you remove or otherwise alter system files (i.e. files that have been installed as part of a distribution) during a shell escape, any previous disk space computations will become invalid. If you may have altered the sizes of system files during a shell escape, you should issue a *recalculate* command to recompute the disk space deltas. There is no harm in recalculating more often than necessary, other than the delay. A reminder is given after each shell escape during which the free disk space changes significantly.

clean	Clears all files and directories from the / and */usr* file systems. All existing files and directories on the *root* and *user* file systems are lost. (See *remove* in the Subsystem Selection menu to remove specific subsystems.) *inst* requests confirmation when a file system already exists on the disk, and does not proceed unless you explicitly answer `yes`. **Caution:** Clean destroys all files on the / and */usr* file systems. Use it with extreme care, only when you want to discard the entire contents of the disk. For peace of mind, back up the entire system before you invoke the clean item. See ''Backing Up and Restoring Your System'' in your owner's guide.
admin [cmd]	Invokes the Administration Functions menu. You can return to the Manual menu when you are finished. If you give arguments, *inst* treats them as an administrative function to be executed immediately, without leaving this menu.
return	Returns to previous menu.
help [item]	The standard *help* item. The default help from this menu is a brief description of the menu and the items on it. For additional information, give the topic you want to find out about as an argument to *help*.
quit	Performs certain cleanup tasks and exits the software installation tool, returning control to IRIX or the *miniroot* special configuration program.

2.4.2 The Subsystem Selection Menu

The Subsystem Selection menu lets you control which subsystems to install on the disk, as well as other supporting tasks. You can examine the list of available subsystems, select or de-select them by name, examine a list of the file names in a subsystem, or remove previously installed subsystems from the disk. Once you remove the unwanted subsystems and make the desired selections, you can start the installation.

To determine whether to install a particular subsystem, consider the need for the functionality provided by the subsystem, and the subsystem's size in relation to the available disk space. The subsystem listing should help you by showing the description of each subsystem, and the change in disk space that results from installing it. You can remove at any time subsystems that have been installed but that you no longer need or want. Also, you can install at a later date any subsystems that you did not install initially. The Subsystem Selection menu contains these items:

list [names]	Lists current subsystem selections. This is the same as the *list* item of the Manual Installation menu; see the complete description above.
files [names]	Lists the names of files in subsystems. By default, lists the names of all files in all subsystems. If subsystem names or patterns are given, *inst* lists only the files in those subsystems.
install [names]	Selects the named subsystems for installation. The names can be

- product, image, or subsystem names
- patterns
- any of the special keywords listed in Table 2–3

Here are some examples:

```
install eoe2.sw.demos
```

Selects the demos subsystem of the eoe2.sw image.

```
install eoe1
```

All of the subsystems in the eoe1 product.

`install eoe1.*.*`

Equivalent; all subsystems in eoe1.

`install eoe[12].sw`

All subsystems in the eoe1.sw and eoe2.sw images.

`install X`

Selects those subsystems for which an older version is installed on the disk. (See Table 2–3)

remove [names]	Selects the named subsystems for removal from your machine.
keep [names]	Cancels any *install* or *remove* selections for the identified subsystems. Use *keep* to identify subsystems that should be kept as is on the disk.

There are several special keywords that you can use to specify the subsystems operated on by the *list*, *install*, *remove*, and *keep* menu items. Most of these keywords also have a one-letter abbreviation that may be used. Note that these one-letter abbreviations are case-sensitive. Table 2–3 lists the keywords and their abbreviations.

Letter	Keyword	Identifies subsystems which:
i	install	are marked for installation
k	keep	are marked for "keep"
r	remove	are marked for removal
I	installed	are already installed
U	uninstalled	are not installed
X	replaces	are replacements for an older installed version
N	replaced	are replaced by a newer installed version
d	default	are declared "default"
c	candidate	are subsystems for which this or an older version could (and may) have been installed at some point in the past (In other words, ''not new'')
(none)	inplace	are declared as being installable without the miniroot

Table 2-3. Keywords

Where applicable, you can combine these keywords (or letters). However, all tests must pass in order for the subsystem to be included. For example, to identify all subsystems that could have been installed at some point in the past but were not, you would use the keywords `candidate uninstalled` (or `c u` together).

default [names]	Sets default selections. The pre-installation checks determine which subsystems to install by default. If you install a product for the very first time, the default subsystems are predefined on the installation tape(s). If you update a product, the default subsystems are those that replace previously installed, corresponding subsystems, plus any pre–defined default subsystems on tapes that you have note installed before. New software directly replaces corresponding subsystems already on the disk.

Note: The *install*, *remove*, *keep* and *default* menu items change the state of subsystem selection as reflected in the subsystem list; they do not start installation. You can review and alter the selections until you are satisfied with them, and then use the *go* item to initiate installation.

step [names]	Selects subsystems using interactive step mode. This item provides a convenient method of traversing the lists. The easiest way to select individual subsystems within a product or image is to use *step*. You can use the following keyword abbreviations in step mode:

i	install; select this subsystem for installation
r	remove; select this subsystem for removal
k	keep; cancel any install or remove request
d	default; use the default operation
p, -	go to the previous subsystem; `<ctrl-p>` has the same effect
n, +	go to the next subsystem; `<enter>` and `<ctrl-n>` have the same effect
/pat	search for a subsystem matching the pattern *pat* and continue from there. This is a

convenient means of starting at a specific subsystem.

l	list subsystems, up to the current subsystem
f	list the file names in the current subsystem
h	display help for step mode
q	quit interactive step mode

step mode also supports the use of the arrow keys:

up-arrow	same as ''p'': go to previous subsystem
down-arrow	same as ''n'': go to next subsystem
left-arrow	same as ''k'': keep this subsystem as is
right-arrow	same as ''i'': select subsystem for installation

Using the `<Shift>` key with any of these keys applies that command to the remainder of the product, rather than just the currently displayed subsystem. (An exception to this is `<Shift>`right-arrow, which is equivalent to `D` rather than `I`.)

Note: The easiest way to select subsystems over large, easily identifiable parts of the product is to use patterns and the *install* or *remove* items. For example, `remove *.man` will select all on-line manual pages for removal.

recalculate

Recalculates the disk space deltas. If you remove or otherwise alter system files (i.e. files that have been installed as part of a distribution) during a shell escape, any previous disk space computations will become invalid. If you may have altered the sizes of system files during a shell escape, you should issue a *recalculate* command to recompute the disk space deltas. There is no harm in recalculating more often than necessary, other than the delay. A reminder is given after each shell escape during which the free disk space changes significantly.

go	Performs the standard pre-installation checks and then installs or removes subsystems based on the current selections.
admin [cmd]	Performs miscellaneous administrative functions. This item invokes the Administrative Functions menu, which is useful for dealing with file systems and network initialization. See "The Administration Functions Menu" below. If you supply arguments, *inst* treats them as an administrative command to be executed immediately, without leaving this menu.
return	Returns to the previous menu from which the Subsystem Selection menu was invoked.
help [keyword]	Displays help information for general or specific subjects. Use *help topics* for a list of available topics and keywords.
quit	Terminates software installation.

2.4.3 The Administration Functions Menu

The Administration Functions Menu contains three kinds of items:

- Items that are hidden on other menus. Such items are included on the administration menu for reference.
- Items that initialize TCP/IP network access.
- Items that work with file systems.

The items related to the network and file systems are useful only in the *miniroot*. The administration menu displays these items:

set [options]	Sets, clears, or lists options. Options take values that are Boolean, integer, or string, depending on their function. Boolean valued functions are set to *on* or *off*. If you give no arguments, *inst* lists the current settings. If you give one argument, it should be the name of an option to be set to the default function. The default for Booleans is always *on*. The default for other types depends on their value. If you supply two arguments, *inst* uses the first as the option name and the second as the new value. For example, the following command sets the number of lines displayed on the screen at a time (i.e., between continuation prompts such as `more`): **`Admin> set lines 24`** Use *help set* to find out about the available list of options and how to use them.
sh [cmd]	Escapes to a shell or immediately runs the given command.

If the free disk space count changes significantly during a shell escape, you will be reminded to use the *recalculate* command if you altered or removed any system files.

shroot [cmd]	Escapes to a chrooted shell or immediately runs the given command, chrooted to the standard file system.
host [name] [addr]	Identifies local host name and/or address, setting the hostname of the system and making the appropriate entries in the *miniroot*'s */etc/hosts* file. If you do not provide a *name*, *inst* prompts for it. If the address of the given name is not in the *miniroot*'s */etc/hosts* file, or in the */etc/hosts* from the normal IRIX file system, *inst* prompts for it.
server [name] [addr]	Identifies remote host name and/or address, making appropriate entries in the *miniroot*'s */etc/hosts* file. If you do not provide *name*, *inst* prompts for it. If the address is not given, and is not in the normal file system's */etc/hosts* file, *inst* prompts for it.
mount [name] [dir]	Mounts an additional file system relative to the normal file system's *root*. By default, all file systems listed in the normal file system's */etc/fstab* are mounted during *miniroot* installations. This item makes it possible to mount other file systems. During *miniroot* installations, all normal file systems are mounted relative to */root*, but are displayed without the */root* prefix.
umount [name]	Unmounts a file system. You can use this item to unmount file systems that were automatically mounted, or were mounted explicitly with the *mount* item.
mkfs [names]	**Warning:** This deletes all files on the named device.

	Makes new file systems on the named devices. By default, the *root* and *user* file systems are remade; this is aliased as *clean* in the Manual Installation menu. You can name a specific device on which to make new file systems. You can then mount the new file system.
versions [options]	Runs the *versions* command. Using *versions remove* at this menu allows you to immediately remove a subsystem. This item is also available from all other *inst* menus as a hidden menu option. See section 2.5, ''Special Features'', for more information on the *versions* command.
return	Returns to the previous menu.
help [item]	The standard help item.
quit	The standard quit item; ends the software installation process.

2.4.4 The Interrupt/Error Menu

The Interrupt/Error menu pops up if you interrupt an installation or if *inst* detects an error. To interrupt the installation, press `<ctrl-c>` unless the interrupt characters have been redefined by the *stty* commands in your configuration files. The menu allows you to:

- stop the installation,
- proceed with the installation, or
- abort the installation, without cleaning up.

If you type `<ctrl-c>` (interrupt) while software is not actually being installed, you terminate operation of the current item. Sometimes, the current operation must be complete before the interrupt is acknowledged, so it might take a moment for the operation to stop. If an error occurs, *inst* displays an appropriate error message before it displays the menu.

Note: While *inst* is executing certain ''auxiliary commands'' it will ignore your `<ctrl-c>`. Some of these procedures, which are clearly identified if you *set verbose on*, can take a few minutes to complete at which time *inst* will allow interruptions to occur again.

The Interrupt/Error menu contains these items:

stop	Stops the installation prematurely, before the next file is installed. (There might be a few moments delay while work on the current file finishes.) Use *stop* if the error condition is such that you cannot or should not proceed with installation.
continue	Proceeds with installation, ignoring the interrupt or error condition. Recommended if the error condition is well understood and known to be innocuous, or if you have simply interrupted the installation to *set verbose* on or off.
help [item]	The standard help command. The default help topic is an explanation of the Interrupt/Error menu.

abort	Immediately terminates *inst*, with absolutely no cleanup actions. This is a rather brutal way of stopping installation, and should be used only in extreme cases where *stop* does not seem to work.

Warning: While installing software, *inst* does not let you use the *sh* or *shroot* items to run the *versions* command or another *inst* because the on-line installation history might become damaged. You can, however, access all of the *versions* functions from the administration menu within *inst*.

2.5 Special Features

inst is the main tool you use to install software. Other tools, such as *versions*(1M) and *distcp*(1M), allow you to do certain types of system and installation administration. This section describes these and other installation tools.

The *distcp* Command

distcp(1M) is an IRIX command that lets you copy a software product from the installation tape(s) to a directory on a remote IRIS-4D Series workstation or server. The word *remote* describes a computer that is connected to your IRIS-4D Series workstation or server over a network.

A remote computer might contain hardware or software that you want to use during the installation. Once a product is in such a directory, you can install it on your system from the remote directory instead of using the cartridge tape on your IRIS-4D Series workstation or server. The 4D1-3.3 version of *distcp* can read all current and older tapes. Earlier versions of *distcp* cannot read 4D1-3.3 version tapes.

Installing software from a remote directory is helpful in situations where many computers must be updated, because network access is generally faster than tape access and more than one computer can access a remote directory at a time.

The following example shows how to copy a product from a cartridge tape in the no-rewind tape device (*/dev/nrtape*) to a directory called *dir*. You must always use the no-rewind tape device with the *distcp* command.

```
distcp /dev/nrtape /dir
```

You must use the 4D1-3.3 version of *distcp* on 4D1-3.3 tapes and images. (4D1-3.3 *distcp* can be used to read 4D1-3.2 tapes and images.)

Since *eoe1* contains the standalone shell *sash* for this release, it should be the last tape read with *distcp*.

Caution: Do not alter any of the files in a remote distribution directory because they might become unusable.

See the *IRIX System Administrator's Reference Manual* for detailed information about tape drive device names and using *distcp*(1M).

The *versions* command

inst maintains an on-line installation history of the products, images, subsystems, and files that are installed or removed from the system during installation. *versions* is an IRIX command that gives you access to that history and lets you perform limited administrative duties. Using *versions*, you can:

- see which subsystems are installed
- list the file names in a product or subsystem
- locate configuration files
- see whether a configuration file was modified since it was installed
- remove subsystems

One common use of *versions* is to determine which configuration files were changed when new software was installed. When you are finished installing your software, enter the following command to identify the configuration files in question.

```
versions changed
```

Use *diff*(1) to compare the configuration files and a text editor to modify the active version. *versions* is available as a menu item from all installation menus, whether it is listed or not. See *versions*(1M) in the *IRIX System Administrator's Reference Manual* for detailed information.

The *lboot* Command

The *lboot*(1M) command allows an experienced system administrator to rebuild a kernel to reflect changes that were made to the operating system or system configuration files. *lboot* configures a bootable kernel using the files in */usr/sysgen/system*, and the files in the directories */usr/sysgen/boot* and */usr/sysgen/master.d*.

Use *lboot* not only after changing a parameter, but also after adding configurable software subsystems, adding software drivers for new hardware devices, or removing software drivers for hardware devices that no longer exist.

To reconfigure the system, first become the superuser, and then follow the procedure below.

1. It is best to save the original kernel (in case, for some reason, you need it later):

   ```
   cp /unix /unix.save
   ```

2. Change your working directory to */usr/sysgen* and run *lboot*, specifying the new kernel as "unix.install":

   ```
   cd /usr/sysgen
   lboot -u /unix.install
   ```

3. Send a message to all users logged into your system that you are bringing it down to install a new kernel. After making sure everybody has logged off, reboot the system:

   ```
   reboot
   ```

   When you run the *lboot* command, the system overwrites the current kernel, */unix*, with the kernel you have just created, */unix.install*. An autoconfiguration script, found in */etc/rc2.d/S95autoconfig*, runs during the startup process. (To override the autoconfiguration, you can rename this file. The *autoconfig* script operates when there have been changes to files in */usr/sysgen* or a new device driver has been installed.)

3. Major Enhancements and Compatibility

This chapter describes the major software and hardware enhancements included in the 4D1-3.3 release, summarizes compatibility issues for the program development tools, and provides a description of the subsystems provided with this release.

This chapter is divided into the following sections:

- Major Enhancements
- Compatibility of Program Development Tools
- Description of 4D1-3.3 Subsystems

3.1 Major Enhancements

This section is divided into two subsections that describe the major software and hardware enhancements (respectively) in release 4D1-3.3. The major software enhancements in the 4D1-3.3 release include:

- IRIX™ enhancements
- New compiler features
- 4Sight™ enhancements

3.1.1 Enhancements to IRIX

Enhancements to IRIX fall into these categories:

- POSIX
- Virtual memory/file systems
- Real-time programming
- Parallel programming
- Logical volumes

  In release 4D1-3.3, IRIX provides the ability to graft several physical disk partitions together into one logical partition, referred to as a *logical volume*. A logical volume can span multiple physical disks, providing both larger data files and greater file system throughput. Improved throughput is the result of increased parallelism in the I/O process by *striping* accesses across several drives and controllers. Using logical volumes, it is possible to create file systems as large as 8 gigabytes and individual data files as large as 2 gigabytes.

  Refer to the manual entries for *mklv*(1M), *lvinit*(1M) and *lvtab*(4) for more detailed information about logical volumes. A step-by-step procedure for creating a logical volume is given in the *IRIS-4D System Administrator's Guide.*

- Extensible file systems

  The IRIX Extent File System™ (EFS) now allows partitions containing EFS file systems to be extended without bringing the system down or rebuilding the existing file system contained in the partition. For example, a logical volume containing an EFS file system can be extended by adding one or more disk partitions to the end of the existing logical volume.

  Note: File systems can only be extended if a logical volume is being created or if there is room remaining in the disk partition (this is usually not the case).

After this has been done, the *growfs*(1M) utility program can be used to extend the existing EFS file system to fill the newly enlarged logical volume. This extension operation automatically preserves all the data in the original file system. It is not necessary to *mkfs*(1M) and then reload the data.

- POSIX compliance

  As a beta site for the NIST POSIX Verification Suite, Silicon Graphics is committed to providing a POSIX-compliant operating system.

  The 4D1-3.3 release provides a major step towards POSIX (IEEE 1003.1) compliance. All functions and interfaces specified by the 1003.1 Full Use Standard are provided with the following exceptions:

  - The behavior of file system operations specified by 1003.1 is guaranteed only for local EFS file systems, not for NFS file systems.
  - The ANSI *setlocale* function as specified and extended by Chapter 8 of the 1003.1 spec is not implemented.
  - The POSIX *tar* and *cpio* program extensions specified in Chapter 10 are not provided.

  Note: At the time release 3.3 was completed, the NIST suite had not yet been released in final form, so Silicon Graphics cannot warrant that release 3.3 will pass the final released version of the NIST POSIX Verification Suite.

- Merged page and buffer caches

  A number of significant changes have been made to the kernel algorithms that manage system memory. All system memory is now treated as a large cache that holds pages of the virtual space of processes and pages from disk files.

  These changes have a number of beneficial effects, including significant performance improvements in the file system and correct semantics when a given file is accessed both as a mapped file and through the traditional *read*/*write* interface.

The cache is subordinate to memory attached to user processes, so caching activity will not affect currently running processes. Any page of free memory, however, is now available to cache disk data, as long as the file's corresponding inode is resident in the in-core inode cache.

- Resident set size limitation

  The kernel page replacement algorithm has been changed to provide the ability to limit the number of physical memory pages that a particular process can use at any given time. This is called the Resident Set Size (RSS) of a process.

  By using this feature, it is possible to minimize the effect that background jobs doing heavy paging have on the interactive response of the system. Refer to the *setrlimit*(2) manual entry for further information.

- No more preallocation of swap space

  In previous versions of IRIX, it was impossible to create any virtual space without having a place to store it (i.e., main memory or swap space). In Release 4D1-3.3, the system has been changed so that storage space is required only when the virtual space actually needs to be stored.

  For example, when a large program calls *fork*(2) and then immediately calls *exec*(2) to execute a smaller program, it is no longer required that the system have enough swap and memory to hold twice the size of the parent process. Since *fork*(2) marks the pages of the parent copy-on-write and most of the parent's pages are never touched by the child before it calls *exec*(2), separate storage space is not required for most of the child's logical pages in the interim between the *fork* and the *exec*.

- Autogrowable mapped files

  In this release it is possible to extend a mapped file by storing to a virtual address corresponding to a location beyond the current end of file. At the time the file is originally mapped using the *mmap*(2) system call, the caller specifies the file offset and length of the segment to be mapped. If the specified segment reaches beyond the end of file and autogrow mode has been requested for the segment, then stores to virtual addresses that correspond to locations beyond the end of file cause the file to be extended automatically to the required length. The extension is done in multiples of pages (4096 bytes each) up to the limit specified by the offset and length given at *mmap*(2) time. All pages added to the file will be zero filled. Refer to the *mmap*(2) manual entry for more information.

- Large virtual spaces handled more efficiently

  The performance of programs that have large virtual address spaces (more than 10 megabytes of code or data) or many virtual address spaces (using many shared libraries or mapped files) is much better in Release 4D1-3.3, as a result of changes in the way the kernel virtual memory code manages the Translation Lookaside Buffer (TLB). The TLB is a 64-entry cache on the microprocessor that caches virtual to physical address translations.

  Large programs that exhibit poor reference locality generate significantly fewer TLB faults in Release 4D1-3.3, resulting in much better performance.

Real-time Programming

The IRIX operating system supports a powerful set of real-time programming features. In combination, a programmer can use these features to time events accurately, use signal handlers as true interrupt routines, control allocation of real memory to the process and provide for priority scheduling.

Real-time programming provides for a range of response time and latency from very fast handling at device interrupt time to high-priority dispatch of user processes to handle the event.

The following enhancements have been made to the real-time features of IRIX in Release 4D1-3.3:

- High-resolution interval timers and time-of-day clock, accurate to 833 microseconds.

  The maximum resolution of the system time-of-day (as returned by the *gettimeofday*(2) call) and interval timers (see *setitimer*(2)) has been increased from 10 milliseconds to 833 microseconds. By default, the resolution is set to 10 milliseconds. If you want the greater resolution, you can enable it as the superuser with the new *ftimer*(1) command.

- Lower interrupt latency

  The 4D1-3.3 release guarantees a maximum interrupt latency of 650 microseconds on properly configured systems. Maximum interrupt latency has been decreased by reducing the worst-case time that interrupts are masked by the operating system. On properly configured systems, worst-case interrupt latency is 650 microseconds (assuming a 25 MHz processor).

- Kernel preemption points have been added to allow non-degrading priority processes to preempt lower priority processes even when the preempted process is executing in the kernel. This provides a significant improvement in worst-case process dispatch latency.

Parallel Programming

- Gang scheduling

  Cooperating processes can now be scheduled as a unit. This ''gang scheduling'' of processes on multiple-processor systems greatly increases the efficiency with which cooperating processes communicate with one another. With gang scheduling it is possible to run parallel jobs efficiently in a time sharing environment. This results in better overall system performance, particularly on systems supporting many simultaneous users or tasks. See *schedctl*(2) for more information.

- Shared graphics within a parallel job

  In Release 4D1-3.3, it is possible for graphics to be a shared attribute within a shared process group. In previous releases, only one process within the share group could be a graphics process. A process that is already a graphics process (that is, has called *winopen*(3G)) and calls *sproc*(2) with a share mask that specifies the sharing of address space automatically creates a shared process group in which graphics is shared. This means that any process within the share group can make Graphics Library calls and that all such calls affect the same window or graphics context.

  It is important to note that Graphics Library code does not prevent separate processes within a parallel job from simultaneously accessing Graphics Library data structures or the graphics pipe. Because the programming model presented by the Graphics Library is fundamentally stateful, the responsibility for the definition and protection of critical regions must be owned by the application program.

  For example, suppose that two processes within the share group want to perform a *bgnpolygon*, *v3f*, *v3f*, ... *endpolygon* sequence, each for a different polygon. As soon as one of the processes has done its *bgnpolygon*, the other process must not touch the pipe until the first process has done the corresponding *endpolygon*. Thus the application code must make the above sequence a critical region in each process in order to ensure that the two processes do not interleave their sequences of calls.

- Improvements to signal handling in parallel jobs

  In Release 4D1-3.3 it is possible to generate a signal to all members of a shared process group when any member of that group dies. The *prctl*(**PR_SETEXITSIG**) system call is used to activate this behavior and to specify the signal that is delivered. Refer to the *prctl*(2) and *sproc*(2) manual entries for details.

- Blocking and unblocking entire shared process groups

  Two new routines are provided that are analogous to *blockproc*(2) and *unblockproc*(2) but that act on entire shared process groups. These are *blockprocall*(2) and *unblockprocall*(2). Refer to the *blockprocall*(2) manual entry for details.

- New debugging support in libmpc

  Two new routines have been added to aid in debugging parallel programs that use the locking primitives in **libmpc**: *usdumplock*(3) and *usdumpsema*(3). These primitives provide the ability to dump the history associated with a single lock or semaphore, respectively.

3.1.2 New Compiler Features

- The 4D1-3.3 release integrates MIPS 2.0 compiler technology.
- The new compilers contain numerous improvements and fixes over the previous release, most notably in the code optimizer.
- Nearly complete ANSI C support:
  - Include files provided with this release have been changed to use ANSI C-type prototypes.
  - ANSI rules and type checking can be turned on with a compiler switch.
  - New ANSI types (for example, **void, void***, **const**) and ANSI constructs are recognized.

For developers who do not need to conform to the ANSI standard immediately, the new compiler permits non-ANSI constructs recognized by the current compiler. If you are porting C applications from Release 4D1-3.2 to 4D1-3.3, be aware that the ANSI C compiler outputs numerous new warning messages where the previous compiler remained silent. If ANSI compatibility is not required for applications you are developing under Release 4D1-3.3, you can ignore these error messages.

For more information on the specific ANSI support introduced in Release 4D1-3.3, read ''Changes to Program Development Tools'' in Chapter 5, ''Changes''.

4Sight™ Enhancements

Release 4D1-3.3 includes a new version of 4Sight (1.5) in which retained canvases are implemented when 4Sight is running in 24-bit RGB mode. Previously retained canvases only worked when the server was running in 8-bit color-index mode. A backgammon program has been included in /usr/NeWS/demos as a demonstration of retained canvases.

Psview(1) no longer caches the current page in 4Sight. This means that the PostScript programs that use the **currentfile** can now be previewed. It also means that 4Sight no longer grows during extensive previewing. A complete and accurate manual page is now provided.

Many bugs have been fixed as described later in these release notes.

3.1.3 Graphics

Sphere Library

Release 4D1-3.3 includes a new subroutine library called the *Sphere Library*, that draws spheres by issuing GL calls. It can be called from C or FORTRAN programs.

The Sphere Library files are */usr/lib/libsphere.a* and */usr/include/gl/sphere.h*. There is an example program located in */usr/people/4Dgifts/src/sphere*.

Appendix B of these Release Notes contains manual pages for the Sphere Library routines. These manual pages are also available on-line.

3.1.4 Hardware Enhancements

The 4D1-3.3 release supports the following new Silicon Graphics hardware:

- PowerVision™ Graphics
- 4D/300 POWER Series systems
- POWER Channel IO controller for POWER Series systems
- IPI2 disk subsystem for POWER Series systems
- System memory expansion to 256 MB on POWER Series systems
- RV2 Raster Subsystem
- Entry Plus Graphics Upgrade (for 4D/20 and 4D/25 systems)
- Multihead Personal IRIS™

PowerVision Graphics

The PowerVision system is a graphics supercomputer that renders 1 million polygons per second and includes hardware support for real-time texture mapping, polygon antialiasing, fog/haze effects, sub-millisecond full-screen clear, arbitrary clipping planes, and image processing operations. It also includes hardware support for near-real-time rendering of higher-quality images, with full-scene antialiasing, depth of field, and motion blur.

Programs compiled with gl_s will run on the VGX workstations without recompilation.

PowerVision is available as a complete system and as an upgrade to existing members of the IRIS Power Series product line.

4D/300 POWER Series Systems

These multiprocessor systems feature CPU boards which have 33 MHz microprocessors, floating point units and associated caches. Each 33 MHz processor board contains two complete CPU and FPU pairs. This board is also available as an upgrade to existing 4D/200 series systems in both server and graphics configurations.

POWER Channel IO Controller

The POWER Channel IO Controller provides a significant increase in IO bandwidth for POWER Series systems. The POWER Channel has the following features:

- Up to 32 megabytes per second VME throughput
- Two complete SCSI controllers, both supporting synchronous SCSI
- Ethernet controller with DMA capability

IPI2 Disk Subsystem

IPI2 is a new high performance VME disk interface. Each IPI2 controller can support up to 8 drives on each of its two independent channels. Each drive has an unformatted capacity of 1.1 gigabytes and a raw transfer rate of 6 megabytes per second. The largest configuration supported at first release will be two controllers, supporting 16 drives each, for a total of 32 gigabytes of high performance disk capacity.

256 MB Memory

Through the use of 4 megabit dynamic RAMs, POWER Series machines now support a maximum memory configuration of 256 megabytes.

RV2 Raster Subsystem

The 4D1-3.3 release includes support for the RV2 raster subsystem. The RV2 is a single board replacement for the RV1/RM1 three-board set.

The GT/GTX systems shipped after 7/1/90 may include RV2 boards and thus require the 4D1-3.3 release.

Note: The RV2 raster subsystem does not contain alpha bitplanes.

Entry Plus Graphics Upgrade

The Entry Plus graphics upgrade allows Z buffered RGB images to be displayed on an eight bit Personal IRIS. This upgrade includes a Z buffer and overlay planes. RGB images are displayed in 8 bits by using dithering to approximate the true color. When RGB mode is enabled in this configuration, double buffering is turned off (because 4 bit RGB mode is not supported).

The *getgdesc* call can be used to get information about the configuration of the current system. In the case of an Entry Plus graphics configuration, *getgdesc* will return the following values:

getgdesc **Inquiry**	**Return Value**
GD_BITS_NORM_SNG_RED	3
GD_BITS_NORM_SNG_GREEN	3
GD_BITS_NORM_SNG_BLUE	2
GD_BITS_NORM_DBL_RED	0
GD_BITS_NORM_DBL_GREEN	0
GD_BITS_NORM_DBL_BLUE	0
GD_BITS_NORM_SNG_CMODE	8
GD_BITS_NORM_DBL_CMODE	4

This information indicates that there are three bits of red, three of green, and two of blue in single buffered RGB mode. It also shows that double buffered RGB mode is not supported.

Multihead Personal IRIS™

For applications that demand multiple screens, the Multihead Personal IRIS is now available. This workstation incorporates up to four multiple Personal IRIS graphics cards, each with their own monitor. The system is based on the IRISette VME card set and is packaged in a Professional IRIS™ chassis.

In this release, only two screens are supported, the second of which is without window management.

3.2 Compatibility of Program Development Tools

This section describes the compatibility between these software releases:

4D1-3.2 and 4D1-3.3

Release 4D1-3.3 is generally compatible with release 4D1-3.2.

3.2.1 Upwards Compatibility

Binaries linked with the non-shared versions of *libfm.a* or *libgl.a* must be relinked. Silicon Graphics recommends always linking with the shared versions of the font manager and the GL.

Most system header files have been changed to conform to the ANSI C specification; in particular, non-ANSI names have been renamed and function prototypes added. As a result, some previously undetected "incorrect" C programs will be detected.

There are a large number of enhancements and changes to Fortran. Please read the Fortran release notes for Fortran specific compatibility information.

In the following table, *source* means source files, whether Fortran, C, or another language. *Application libraries* means not only archive libraries of object files (*.o* files), but also individual object files. *Executables* means executable binaries (not shell command files).

	4D1-3.3 System Software		
4D1-3.2 Generated	Compilation Environment (cc, f77, ar, ld)	Tools Environment (nm, strip, dbx, edge, lint, stdump, size, dis)	Execution Environment
source	yes	NA	NA
application libraries	yes	yes	NA
executables	NA	yes	yes, unless the non-shared versions of the font manager or GL were used

3.2.2 Backwards Compatibility

While object files and executables created under 4D1-3.3 may work on an older system, this use is not recommended. Indeed, it is dangerous to run a new executable on an older system as it may have unpredictable consequences.

The 3.2 loader, *ld*, will generate warning messages if it encounters an object file generated on a 3.3 system. This warning can be ignored.

Due to changes in the shared library format, executables linked under 3.3 to shared libraries will not execute correctly under 3.2 or older systems. And, of course, executables linked to private libraries under 3.3 should not be executed on 3.2 or older systems.

3.3 Description of 4D1-3.3 Subsystems

Execution Only Environment Tape 1

eoe1.sw.unix	This subsystem contains the core IRIX commands and files and is required on all systems.
eoe1.man.relnotes	This is an on-line readable copy of the IRIX release notes.
eoe1.man.unix	This is an on-line copy of the IRIX user's manual. Entries may be viewed with the ''man'' command.

Execution Only Environment Tape 2

eoe2.sw.NeWS	This is the 4Sight window manager and is required by all workstations. This subsystem contains the "standard" NeWS fonts.

```
Courier                   Icon
Courier-Bold              Iris
Courier-BoldOblique       Screen
Courier-Oblique           Screen-Bold
Cursor                    Times-Bold
Helvetica                 Times-BoldItalic
Helvetica-Bold            Times-Italic
Helvetica-BoldOblique     Times-Roman
Helvetica-Oblique         type
```

eoe2.sw.X11	This is the run-time X11 windowing package. It is required to run X11 applications. This subsystem includes the following fonts that are required to run X, along with some terminal emulator fonts expected by xterm users:

```
6x10    cursor
6x12    Terminal
6x13    Terminal-Bold
8x13    Terminal-BoldNormal
8x13b   TerminalNormal
9x15
```

eoe2.sw.Xapps	These are X client applications.
eoe2.sw.Xdemos	These are demonstration programs that make use of the X windowing system. You must install eoe2.sw.X11 in order for these programs to work.

eoe2.sw.Xfonts	These are a set of fonts which were shipped with X11 Release 2. They are no longer a part of X, but some older X programs require these fonts to be in place.

```
apl              German              sup
arrow            Greek               supsup
chess            Hebrew              swd
chp              ipa                 sym
cyr              krivo               vbee
Cyrillic         lat                 vctl
dancer           met                 vg
fcor             micro               vgb
fg               mit                 vgbc
fg-Bold          oldera              vgh
fg-Oblique       plunk               vgi
fgone            rot                 vgvb
fgone-Bold       runlen              vgl
fgone-Oblique    sansserif           vmic
fgs              sansserif-Bold      vply
fqxb             sansserif-Oblique   vr
fr               serif               vrb
fr-Bold          serifb              vri
fr-Oblique       serifi              vsg
frone            stan                vsgn
frone-Oblique    stempl              vshd
frthree          sub                 vxms
frtwo            subsub              xif
```

eoe2.sw.acct	This is the System V Process Accounting package. It is used to monitor system resource usage on a per-user basis. When Process Accounting is installed and turned on, a record is kept of every command that is executed along with data on the resources the command used. Report generating scripts are then run which produce periodic reports of system utilization. This package is useful when it is desired to monitor usage patterns in systems with large numbers of users.

eoe2.sw.bsdlpr	This subsystem contains 4.3BSD line printer software to allow the IRIS to access remote printers using the *lpd* protocol. Using the lpd daemon to control local printers is not supported.
eoe2.sw.cdsio	This is the driver for the VME Serial Expansion board. It is required for systems with that option installed.
eoe2.sw.crypt	This subsystem contains the *crypt* and *makekey* programs which, along with the editors *ed* and *vi*, implement a simple text encryption system.
eoe2.sw.demos	This is the standard SGI demonstration package. It has been enhanced and expanded with this release and features a menu driven front end called ''buttonfly''. Some programs may not image properly on machines with eight bitplanes or without a Z buffer.
eoe2.sw.dfm	This is a subset of IRIX utilities dealing with directory and file management. It is generally required in all systems.
eoe2.sw.editors	This subsystem contains the standard IRIX text editors and is generally required on all systems.
eoe2.sw.envm	This is the IRIX WorkSpace package, a user-friendly alternative to the standard IRIX command shell.

eoe2.sw.gltools

This subsystem contains a collection of simple tools that perform a variety of graphics-related functions on a workstation. Complete source code for these tools is contained in the dev.sw.giftssrc subsystem described in the next section. These tools include:

```
blanktime    imged              mag
cedit        icut               mousewarp
clock        imgexp             savemap
dialwarp     interp scrsave
gamcal       ipaste             showmap
gamma        istat              snapshot
ical         loadmap            textcolors
```

eoe2.sw.hyper

This is the driver for the HyperNet VME interface and is required only on systems with that option installed.

eoe2.sw.ikc

This is the driver for the VME Parallel Expansion interface and is required only by systems with that option installed.

eoe2.sw.ipc

This subsystem contains utilities useful in controlling the System V Inter Process Communications facility. It consists of two utilities: *ipcrm* and *ipcs*.

eoe2.sw.ipgate

This subsystem contains additional network routing daemons for machines with multiple network interfaces (gateways). The *gated* routing daemon supports RIP, EGP and HELLO routing protocols. The *mrouted* allows forwarding of IP multicast packets between networks.

eoe2.sw.lp

This subsystem contains the Line Printer Spooling package and is required on any system that will be used with printers, either locally or over a network.

eoe2.sw.mast

This subsystem contains vital graphics software and is generally required on workstations. On some systems, this subsystem will be empty however.

eoe2.sw.moregltools

This is an extension of the eoe2.sw.gltools subsystem and contains commands that implement simple image

	processing functions. The complete source to these commands is contained in the def.sw.giftssrc subsystem described in the next section.
eoe2.sw.optfonts	These are additional fonts that are not required by the base system but may be desirable for some applications.

```
Boston                  NewCenturySchoolbook-Bold
Charter-Black           NewCenturySchoolbook-BoldItalic
Charter-BlackItalic     NewCenturySchoolbook-Italic
Charter-Italic          NewCenturySchoolbook-Roman
Charter-Roman           Symbol
Kanji
```

eoe2.sw.perf	This subsystem contains the System Activity Reporting (SAR) package. This facility is useful for monitoring processor activity and IRIX system performance. It may be used in either an interactive mode or as a background data collector/report generator. This subsystem is not required, but may be useful in diagnosing system performance problems.
eoe2.sw.spaceball	This subsystem provides IRISphere™ support.
eoe2.sw.spell	This subsystem contains a dictionary and command that checks spelling.
eoe2.sw.sysadm	This is a system administration package that does not require graphics and is intended for use primarily on server machines.
eoe2.sw.tcp	This subsystem contains programs and files that implement the TCP/IP family of networking facilities. It is generally required on all systems even if they will be used in a stand-alone environment.
eoe2.sw.terminf	This is the Terminfo terminal database. It contains files describing the capabilities of hundreds of different types of terminals and is used by the *vi* editor as well as many common terminal-oriented

	applications. A few common terminals may be used without this subsystem.
eoe2.sw.ts	This is the driver for the ISI VME quarter-inch cartridge tape controller. It is required on systems with that option installed.
eoe2.sw.uds	This subsystem contains support for 4.3BSD UNIX domain sockets if you install eoe2.sw.tcp. This subsystem is required if you install the eoe2.sw.bsdlpr subsystem.
eoe2.sw.usrenv	This subsystem contains a subset of the standard IRIX commands and is generally required on all systems.
eoe2.sw.uucp	This is the traditional UUCP communications package which implements a point-to-point networking facility. This is only required for sites where UUCP is used but it also contains facilities that may be useful for systems with modems.
eoe2.sw.vadmin	This is a graphical interface to the standard IRIX administration utilities. It provides user-friendly tools for managing printers, users, disks, networks and other common administrative function.
eoe2.sw.xm	This is the driver for the Xylogics VME 1/2" tape controller and is required only on systems with the controller installed.
eoe2.sw.xylsio	This provides 16-port serial board support.
eoe2.man.X11	This is an on-line version of the *IRIX User's Reference* entries that describe the commands in the eoe2.sw.X11 subsystem.
eoe2.man.bsdlpr	These are man pages for the BSD lpd/lpr line printer utilities.
eoe2.man.demos	This is an on-line version of the *IRIX User's Reference*, section 6, which describes the demonstration programs.
eoe2.man.spaceball	These are man pages that support the IRISphere product.

Development Option

dev.sw.G0libraries	This subsystem contains versions of all of the standard programming libraries compiled with the **–G 0** option. These versions of the libraries are generally not required.
dev.sw.NeWSimg	This is an image database for the 4Sight image demonstration programs.
dev.sw.bsdhdrs	This subsystem contains symbolic links for BSD header files formerly in */usr/include/bsd.* In Release 3.3, BSD header files are in */usr/include.* This subsystem is needed if you are upgrading from previous releases or you have source code that include BSD headers with explicit mention of the bsd subdirectory, for example: `#include <bsd/foo.h>`
dev.sw.cc	This subsystem is contains all of the standard IRIX commands and files for compiling and debugging C programs.
dev.sw.cedgetut	This contains files that accompany documentation on how to use *Edge*, the IRIX window-based debugger, when programming in C.
dev.sw.crypt	This subsystem contains the file *libcrypt.a* for use by programs that perform data encryption.
dev.sw.debug	This contains the IRIX kernel debugger and is useful only to those developing kernel device drivers.
dev.sw.giftssrc	This subsystem contains a multitude of sample programs in source code form including the source for all of the eoe2.sw.gltools and eoe2.sw.moregltools subsystems described in the previous section. While the installation of this subsystem is not required, many developers have found the example programs to be extremely useful in learning about the GL, networking, and generic SCSI interfacing.

dev.sw.moredemos	This is an extension of the eoe2.sw.demos subsystem and contains more demonstration programs. These programs are segregated here due to disk space requirements.
dev.sw.rcs	This contains the Revision Control System (RCS) which is a set of programs that may be used to control a source code development project. With RCS, changes to source files are kept in a database with comments such that previous versions of a particular file may be retrieved.
dev.sw.sccs	This contains the Source Code Control System which is identical in purpose although different in use to RCS (described above).
dev.man.cc	This contains the *IRIX Programmer's Reference* manual in an on-line readable format.

3.4 Subsystem Sizes

This is a list of all the subsystems and their sizes. *Default Install* indicates subsystems that are are installed by default when you install using *automatic* mode, or you select *default.* To install subsystems that are not installed by default, you must select them with manual installation features. All sizes are in kilobytes.

subsystem	Personal IRIS	IRIS 4D & Power Series	Data Station	Power Vision	Default Install
eoe1.sw.unix	16248	15085	12081	11796	yes
eoe1.man.relnotes	185	185	185	185	yes
eoe1.man.unix	2234	2234	2234	2234	yes

Table 3-1. S4-EOE1-3.3

subsystem	Personal IRIS	IRIS 4D & Power Series	Data Station	Power Vision	Default Install
eoe2.sw.NeWS	6565	6565	5949	6565	yes
eoe2.sw.X11	2192	2192	2192	2192	no
eoe2.sw.xapps	11602	11602	11602	11602	no
eoe2.sw.Xdemos	1444	1444	1444	1444	no
eoe2.sw.Xfonts	2368	2368	2368	2368	no
eoe2.sw.acct	468	468	468	468	no
eoe2.sw.bsdlpr	550	550	550	550	no
eoe2.sw.cdsio	23	23	0	0	yes
eoe2.sw.crypt	29	29	29	29	yes
eoe2.sw.demos	2385	2274	1385	2277	yes
eoe2.sw.dfm	499	499	499	499	yes
eoe2.sw.editors	385	385	385	385	yes
eoe2.sw.envm	3435	3435	3435	3435	yes
eoe2.sw.gltools	425	425	0	425	yes
eoe2.sw.hyper	99	99	99	99	no
eoe2.sw.ikc	15	15	0	0	yes
eoe2.sw.ipc	121	121	121	121	yes
eoe2.sw.ipgate	334	334	334	334	no
eoe2.sw.lp	759	759	759	759	yes
eoe2.sw.mast	0	490	0	1528	yes
eoe2.sw.moregltools	1307	1307	1307	1307	no
eoe2.sw.optfonts	2273	2273	2273	2273	no
eoe2.sw.perf	802	802	782	782	yes
eoe2.sw.spaceball	399	399	399	399	no
eoe2.sw.spell	433	433	433	433	no
eoe2.sw.sysadm	260	260	260	260	yes
eoe2.sw.tcp	4236	4236	4236	4236	yes
eoe2.sw.terminf	956	956	956	956	no
eoe2.sw.ts	15	15	15	15	no
eoe2.sw.uds	18	18	18	18	yes
eoe2.sw.usrenv	526	526	526	526	yes
eoe2.sw.uucp	1126	1126	1126	1126	no
eoe2.sw.vadmin	4425	4426	4426	4426	yes
eoe2.sw.xm	31	31	31	31	no
eoe2.sw.xylsio	28	28	0	0	no
eoe2.man.X11	563	563	563	563	no
eoe2.man.bsdlpr	23	23	23	23	no
eoe2.man.demos	164	164	164	164	yes
eoe2.man.spaceball	10	10	10	10	no

Table 3-2. S4-EOE2-3.3

subsystem	Personal IRIS	IRIS 4D & Power Series	Data Station	Power Vision	Default Install
dev.sw.G0libraries	5038	5038	3731	3731	no
dev.sw.NEWSimg	6360	6360	6360	6360	no
dev.sw.bsdhdrs	0	0	0	0	no
dev.sw.cc	10607	10619	9292	8995	yes
dev.sw.cedgetut	8	8	8	8	no
dev.sw.crypt	15	15	15	15	yes
dev.sw.debug	433	329	157	157	no
dev.sw.giftssrc	3548	3548	3548	3548	no
dev.sw.moredemos	3165	5675	2967	9634	no
dev.sw.rcs	839	839	839	839	no
dev.sw.sccs	845	845	845	845	no
dev.man.cc	2062	2062	2062	2062	yes

Table 3-3. S5-DV01-3.3

4. Additions

This chapter describes additions to IRIX, the Graphics Library, and demos in the 4D1-3.3 software release.

4.1 Additions to IRIX

4.1.1 BSD Compatibility

Developers who are porting BSD–based applications to 4D1-3.3 will find the transition to IRIX easier with this release. Release 4D1-3.3 provides full 4.3BSD signal support as well as BSD header file and library compatibility.

New BSD features supported in Release 4D1-3.3 include:

- per-process multiple group IDs
  - — The default under IRIX is that a process has one group ID but BSD multiple group mode can be activated with the *multgrps(1)* command.
- the *sigvec*(3B) routine and all the associated calls for specifying signal handling
- the *setrlimit*(2) system call for limiting process resource consumption
- UNIX domain sockets
- the BSD line printer spooler (*lpr* and *lpd*)

IRIX now includes a version of the Berkeley *lpr/lpd* print spooler system. All of the Berkeley spooler commands are available, together with their manual entries. This software allows IRIS-4D systems that are configured in a heterogeneous network to send print requests to BSD print servers.

Note: The functionality of this printer spooler is limited to the functionality provided in the 4.3BSD system. In particular, there is no support for PostScript® printers. For this reason, it is not supported for use as the primary native spooler system for IRIS-4D systems.

- the 4.3BSD versions of *diff*(1), *chown*(1), and *chgrp*(1). These commands are a superset of the System V versions.
- the *renice*(1), *w*(1), *uptime*(1), and *whoami*(1) programs.
- the 4.3BSD *libc* routines *setlinebuf*(3), *setbuffer*(3), *vsyslog*(3), *getrusage*(3), and *vfork*(2) have been added. *vfork* is emulated using *sproc*(2) and *prctl*(2).

BSD Header File Relocation

In this release, BSD header files formerly in the */usr/include/bsd* subdirectory are now in */usr/include*. Therefore, you no longer need the *cc*(1) directive **–I/usr/include/bsd**. For compatibility with previous IRIX releases, the *dev.sw.bsdhdrs* subsystem contains symbolic links for files that used to reside in */usr/include/bsd*, linking them to their new locations. If you had (non-portable) source code that included BSD headers with the explicit *bsd/* prefixes in the `#include` statement, it is worthwhile to remove the prefix for portability. Future releases of IRIX may not provide these links.

BSD Library Compatibility

Most of the routines in */usr/lib/libbsd.a* are now in the standard C library. These include routines such as *gethostbyname*(3N), *rcmd*(3N), and *ndbm*(3). Therefore, it is not necessary (in most cases) to use the *cc*(1) directive **–lbsd** when compiling and linking BSD programs.

The remaining routines in *libbsd.a* have the same names as standard C library routines, but have different invocation syntax or semantics. These routines include:

getgroups (3B)
initgroups (3B)
setgroups (3B)
getpgrp (3B)
setpgrp (3B)
chown (3B)
fchown (3B)
dup2 (3B)

The BSD versions of these routines are also in the standard C library with the prefix *BSD* (for example, *BSDchown*) in case you need to ''mix and match'' System V and BSD routines.

Note that in previous releases there was no *setpgrp* routine in *libbsd.a*. Thus, a program that was linked with **–lbsd** and called *setpgrp* was actually calling the System V version of *setpgrp* in previous releases, but will call the BSD version when compiled the same way under Release 4D1-3.3. Since the syntax of the two versions of *setpgrp* is different, this change may cause some porting problems.

Refer to the *intro*(3) manual entry for more details about the procedure for compiling and linking BSD compatible programs.

4.1.2 File System Additions

As part of supporting the BSD/POSIX multiple group feature, Release 4D1-3.3 IRIX also provides the ability to give BSD group ownership behavior when creating files.

When a process creates a file in a System V.3 system, the owner of the new file is the effective user id of the process creating the file and the group of the file is set to the effective group ID of the creating process.

In a BSD system, the owner of a new file is set to the effective user ID of the process creating the file, but the group of the file is set to the group of the containing directory.

The default behavior in Release 4D1-3.3 IRIX is the System V.3 behavior. You can specify the BSD behavior in two different ways: on an individual directory or on every directory in an entire file system.

To specify the BSD behavior for all files created within a single directory, turn on the `set group ID` permission bit for that directory. You can do this with the following command:

```
chmod g+s directory_name
```

To activate the BSD behavior on an entire file system, set the **grpid** mount option on that file system at mount time by specifying the **grpid** mount option in the */etc/fstab* entry for that file system. The following *fstab* entry does this:

```
/dev/usr /usr efs rw,raw=/dev/rusr,grpid 0 0
```

Refer to the *chmod*(1) and *fstab*(4) manual entries for further information.

Sticky Directories

In a standard System V.3 UNIX system, a user has the ability to delete a file if (and only if) the user has write permission on the directory that contains the file. There are some directories in a system (for example, */tmp* and */usr/tmp*) that need to be writable by all users on a given system. With the standard System V.3 permission rules, this gives any user the ability to delete files belonging to other users. For example, any user on the system could issue the command `rm /tmp/*`, deleting everything in */tmp*.

In order to provide the ability to prevent this, Release 4D1-3.3 IRIX provides a new feature adopted from other variants of UNIX: turning on the *sticky* mode bit on a directory changes the rules for deleting files from that directory. A user can delete a file in a *sticky* directory if and only if the user has write permission on the directory and at least one of the following is true:

— the user is the superuser

— the user owns the directory

— the user owns the file to be deleted

— the user has write permission on the file to be deleted

Set the sticky bit with the command:

```
chmod +t /tmp
```

A directory with the sticky bit set can be recognized from the output of *ls –l* by the `t` in the world execute permission position of the modes field:

```
drwxrwxrwt   3 sys      sys          512 May 10 20:46 /tmp
```

Note that if the sticky bit is set on a directory, the permission checking rule given above applies to any operation that would result in a file being removed, not just a *rm*(1) command. If */tmp* is a sticky directory, then the command

```
mv foo /tmp/bar
```

will fail unless the executing user has the ability to delete the file */tmp/bar* according to the rule given above.

Every time the system is booted into multiuser mode, the script */etc/rc2.d/S58RMTMPFILES* is executed. This script deletes everything in the */tmp* directory and creates the directory */usr/tmp* if it doesn't already exist, among other functions. In Release 4D1-3.3, this script has been changed to set the sticky bits on */tmp* and */usr/tmp* to enable the more restrictive file deletion semantics described above.

If you prefer the previous (less restrictive) behavior, modify the script by finding the line

```
chmod 1777 /tmp /usr/tmp
```

and changing it so that it does not turn on the sticky bit:

```
chmod 777 /tmp /usr/tmp
```

4.1.3 New Commands and Utilities

- The new *passmgmt*(1M) command allows the superuser to update */etc/passwd* entries.
- The new *unifdef*(1) command automatically strips or reduces `#ifdef` constructs in C source or any source filtered by the C preprocessor, *cpp*(1).
- Keyboard auto-repeat rate may be adjusted on workstations using the new *keywarp*(6) command.
- The new –A option to *ls*(1) lists entries beginning with a dot except for the . and .. directories.
- The new –s option to *hostname*(1) trims any domain information from the printed name.
- The new –f option to *which*(1) uses the current path by ignoring the *.cshrc* file.
- *login*(1) has several new options to require passwords, record and display last login times, and *syslog*(3) failed and successful network or tty login attempts. See the *login*(1) manual entry for details.

4.1.4 SCSI Driver

There is a new feature in the SCSI driver that can be used when devices must be connected that either do not support the disconnect feature or do not work correctly when it is enabled. This feature may be enabled by editing the file */usr/sysgen/master.d/scsi* and changing the value of the variable `scsi_enable_disconnect` from `1` to `0`.

Enabling this feature (setting the variable to `0`) may impact performance somewhat when multiple devices are on the SCSI bus, since operations on different devices will no longer be overlapped. Also note that when this feature is enabled, some Exabyte tape drives may not work correctly unless switch #6 is set.

4.2 Additions to the Graphics Library

- A new routine, *frontface*, controls the removal of frontfacing polygons. Its function is similar to *backface*, except that it discards polygons whose vertices are in the opposite order. See the *frontface*(3G) manual entry for more information.
- *lmcolor* is now functional on IRIS-4D B and G models.

4.2.1 PowerVision Commands

The 4D1-3.3 release introduces PowerVision, a new high-performance graphics architecture. Many new GL commands have been added and several existing routines have additional functionality on PowerVision systems.

New Routines

The following list describes the new Graphics Library routines available for use with PowerVision. These routines are explained in more detail in the *Graphics Library Programmer's Guide* and the *Graphics Library Reference Manual.*

acbuf	operates on the accumulation buffer
acsize	specifies the number of bitplanes per color component in the accumulation buffer
afunction	specifies alpha test function
bgnqstrip	begins a quadrilateral strip vertexes

clipplane	specifies a plane against which all geometry is clipped
endqstrip	ends a quadrilateral
fogvertex	specifies fog density for per-vertex atmospheric effects
getscrbox	reads back the current computed screen bounding box
mswapbuffers	swaps multiple framebuffers simultaneously
nmode	specifies renormalization of normals
pixmode	specifies pixel transfer mode parameters
polymode	controls the rendering of polygons
polysmooth	specifies antialiasing of polygons
sclear	clears the stencil bitplanes to a specified value
scrbox	controls the computation of a screen space bounding box
scrsubdivide	subdivides lines and polygons to a screen-space limit
stencil	alters the operating parameters of the stencil
stensize	specifies the number of bitplanes to be used as stencil planes
swritemask	specifies which stencil bitplanes can be written
t	specifies a texture coordinate
tevbind	selects a texture previously defined with *texdef(2D)* to be used for current drawing
tevdef	defines a texture mapping environment
texbind	attaches a predefined texture to one of several system textures
texdef2d	converts a 2-dimensional image into a texture
texgen	specifies automatic generation of texture coordinates

Changes to Existing Routines

- The *overlay* and *underlay* routines has additional functionality under PowerVision. In earlier GL releases, calling *overlay* with a non-zero argument forced *underlay* to zero. Because PowerVision supports simultaneous *overlay* and *underlay*, it simply allocates resources based on the sequence of *overlay*, *underlay*, *singlebuffer*, and *doublebuffer* calls, giving priority based on the calling order (specifically, the last call wins).

  The IRIS PowerVision supports allocation of 8 bitplanes, either single or double buffered, to *overlay* or *underlay*. When this allocation is requested, the alpha bitplanes (which must be present) are used, and are therefore not available for storage of destination alpha values (see *blendfunction*).

  Single or double buffer operation is specified by calling either *singlebuffer* or *doublebuffer* while in drawmode OVERDRAW or UNDERDRAW, prior to calling *gconfig*.

  Currently only *colormap* mode is available in the PowerVision overlay and underlay framebuffers. Double buffered overlay and underlay are swapped when *mswapbuffers* is called with the OVERDRAW or UNDERDRAW bit set.

- *drawmode*

  The values of the *drawmode* tokens have changed. Old values are still accepted by *drawmode* for compatibility. *getdrawmode*() will return the new values. Programs should always use the symbolic constants so they don't have to depend on the values of these constants.

- *mmode*

  A new matrix mode parameter value, MTEXTURE, has been added to *mmode* for texture mapping on PowerVision systems. See *mmode*(3G) for information on the use of MTEXTURE. The texture matrix is also initialized to the identity matrix.

- *readsource* is always SRC_AUTO when not in NORMALDRAW drawmode.

PowerVision Compatibility with Other Systems

This section lists various compatibility issues between PowerVision (VGX) and GT/GTX systems. Several Graphics Library routines exhibit different functionality on these two types of systems.

- *subpixel*

  By default *subpixel* mode is FALSE. GT/GTX systems achieve their rated performance in this default mode. On VGX systems, however, performance is substantially degraded when *subpixel* is FALSE. The published performance figures for VGX machines are valid only when *subpixel* mode is explicitly set to TRUE. Because the scan conversion of lines is affected by *subpixel* mode, some 2-D orthographic programs may require *subpixel* FALSE while drawing lines. Refer to the *subpixel* man page for details.

- *move/draw*

  Lines drawn using the *move/draw* sequence are forced to be flatshaded on the G and GT/GTX models. However, both the Personal Iris and the VGX systems shade lines according to the *shademodel* argument.

- *rmv/rdr*

  Matrix operations are not supported between *rmv/rdr*, *move/draw* sequences.

- *zwritemask*

  The argument to *zwritemask* has always been documented to be a mask, however the GT/GTX and Personal Iris interpreted it as a flag. The VGX interprets the argument as a true mask. Therefore, *zwritemask*(TRUE) will mask the zbuffer to one bit on VGX systems.

- *old style polygons/circf/arcf*

  By default, the GL renders filled circles and arcs as old styled polygons. Currently, on the VGX systems, the filling algorithm of these polygons has problems with missing pixels. To work around this, one can call glcompat(GLC_OLDPOLYGON, 0) prior to drawing filled circles, arcs and old styled polygon drawing commands. However, using old styled polygon commands to draw screen aligned rectangles works perfect. Therefore, any application that uses this feature will continue to works. An example of this is the window manager 4sight.

- *rot accuracy*

  The rot routine on the VGX systems is more accurate than the GT/GTX systems.

- *gversion*

  Some programs that look for specific return values from *gversion* quit if they don't see the GT/GTX values. Such programs fail to work on VGX machines.

  In general, use of *getgdesc* to query for specific capabilities should replace all uses of *gversion* because this behavior will be forward-compatible.

- *concave decomposition*

  On GT/GTX systems, concave decomposition is always on. On Personal IRIS and PowerVision systems, however, concave decomposition is off by default. You must issue a concave(TRUE) call to instruct the system to expect concave polygons. See the *concave*(3G) manual entry for more information.

Known User Errors

This section lists various common user errors that we have found.

- *lmbind/MSINGLE*

  lmbind() can not be used in the MSINGLE mode. The lmbind manual page notes this. However, we did find some applications that did this.

- *zbuffer/czclear*

  On the GT/GTX machines, the zbuffer value is represented as a 23 bit unsigned integer. This meant that the getgdesc of GD_ZMIN is 0. The GD_ZMIN value is typically used for lsetdepth() and czclear(). However, some applications used czclear with z hardcoded 0 for the z value. This will break on the VGX and Personal Iris machines as GD_ZMIN is 0xff800000, or a large negative value.

4.2.2 New Display List Commands

The Graphics Library included with the 4D1-3.3 release adds display list support for high-performance graphics commands previously restricted to immediate execution. This capability was achieved without a significant impact on immediate-mode performance.

The Graphics Library's implementation of graphical objects (see Chapter 16 of the *Graphics Library Programmer's Guide*) calls for GL drawing primitives to behave differently depending on whether the GL is in immediate-mode or object-creation mode at the time a primitive is invoked. Previous implementations of the GL included conditional branches within each GL routine that selected between immediate-drawing and object-creation behaviors. This implementation precluded the use of the high-performance drawing routines in graphical objects because the requisite conditional branching reduced performance.

Beginning with Release 4D1-3.3, the selection between immediate-mode and object-creation behaviors is implemented by switching shared-library jump tables when entering and leaving object-creation mode. This implementation allows the high-performance drawing routines to perform well enough that they can be used within graphical objects.

The following routines are now available within graphical objects:

bgnpoint, endpoint
bgnline, endline
bgnclosedline, endclosedline
bgnpolygon, endpolygon
bgntmesh, endtmesh, swaptmesh

n3f

c3s, c3i, c3f
c4s, c4i, c4f

v2s, v2i, v2f, v2d
v3s, v3i, v3f, v3d
v4s, v4i, v4f, v4d

cpack
wmpack

czclear

linesmooth, pntsmooth, subpixel

In earlier releases, executing these routines in object-creation mode was strictly prohibited by the manual. If a program called them, they did immediate drawing, as if the GL were in immediate mode. Beginning with Release 4D1-3.3, calling these routines in object-creation mode results only in building the current object; no immediate-mode drawing is done.

The ability to use these routines within graphical objects comes at the price of slightly lower performance for the routines that switch the GL into and out of object-creation mode: *makeobj*, *closeobj*, and *editobj*.

This implementation depends on the use of shared-library jump tables, which are not available to the non-shared version of the GL. In order to make the non-shared version function identically to the shared-library version, its versions of the high-performance drawing routines must use conditional branching, and they are thus somewhat slower than they were before.

Use of the shared GL is strongly encouraged for a number of reasons; this is yet another reason to use it instead of the older, non-shared version. To use the shared-library version, use `-lgl_s` instead of `-lgl` when compiling.

4.2.3 NURBS

The NURBS interface to the Graphics Library includes these enhancements in Release 4D1-3.3:

- 8th-order curve rendering on the Personal IRIS
- unlighted rendering on all IRIS-4D platforms
- new surface capabilities

  In previous releases, users could describe only the geometry of a trimmed NURBS surface and render it. With this release they may also specify texture coordinates and color coordinates in addition to positional coordinates. This information is communicated through the use of the type field in the call to *nurbssurface*.

  Specifying texture and color surface information has an effect only on VGX machines. On IRIS-4D G, GT, GTX and Personal Iris, such specifications have no effect.

4.2.4 Graphics Input Queue

A new GL subroutine *qgetfd*(3G) allows a GL program to use the *select*(2) system call to determine when there are events waiting to be read in the graphics input queue. A call to *qgetfd* returns a file descriptor that may be used as part of the *readfds* parameter of the *select* system call. When *select* indicates that the file descriptor associated with the graphics input queue is ready for reading, a call to *qread*(3G) or *blkqread*(3G) will not cause the program to block. See the description of *select* in the *IRIX Programmer's Reference Manual* for more information.

A new GL subroutine *qcontrol*(3G) provides an interface for various administrative functions in the graphics input sub-system. With *qcontrol*, it is possible to add new types of input devices and programmatically control the keyboard auto-repeat rate and mouse acceleration parameters.

4.2.5 Multiple Screens

As of this release, the Graphics Library supports applications opening windows on more than one screen. The routines added to provide this support are:

- *scrnselect* selects the screen upon which windows are opened.
- *scrnattach* attaches the input focus to a specified screen.
- *getwscrn* returns the screen upon which the current window is displayed.

For more information, see the *4Sight Programmer's Guide* and the manual entries for the individual routines.

4.3 Additions to Demos

Release 4D1-3.3 includes these new demos:

Demo	Description
trimnurbs	trimmed NURB (non-uniform rational B-spline) surfaces
revolve	interactive surface of revolution editor

5. Changes

This chapter describes changes to

- IRIX
- networking
- documentation
- graphics
- demos
- program development tools
- software installation tools
- 4Dgifts
- IRIS WorkSpace
- 4Sight

5.1 Changes to IRIX

A number of changes to the file system buffer cache and virtual memory have eliminated many operating system tuneable parameters, and removed the necessity of changing others. Please consult the *System Configuration and Tuning Guide* for more details.

The 4D1-3.3 release includes these changes to IRIX:

- The header files in */usr/include/bsd* and its subdirectories have been moved to */usr/include*. If you had old source code that incorrectly included BSD headers files with

  ```
  #include <bsd/foo.h>
  ```

  or you are upgrading from a previous IRIX release, install the *dev.sw.bsdhdrs* subsystem. This subsystem uses symbolic links to provide compatibility. It will not be supported in the next major release.

- *rcsdiff*(1) now understands the new *diff*(1) **–biwt** options. The **–q** (quiet) option returns *diff*'s exit status.

- Several additions have been made to the BSD *Mail* program (*mail_bsd*(1)):

  — lines starting with # are ignored

  — new "unread" command

  — better handling of RFC822-style addresses, and ability to add all 822 header fields via tilde escape commands

  — support for "reply-to" header field

  — uses the TIOCGWINSZ ioctl call to determine the actual screen size

  — fixed mailfile locking to be compatible with the System V (mail) file locking scheme

  — added read-only mailfile locking to prevent multiple instances of Mail from having simultaneous write access to the mailfile

  — applied *flock*(3)'ing to all types of lock files so that the lock scheme works over NFS

- System V mail (*mail_att*(1)) now supports "reply-to" and uses *flock*(3) for locking files, like *Mail*.

- You can change *savecore*(1M)'s default location for storing kernel crash files with */etc/config/savecore.options*.

- The system logging daemon, *syslogd*(1M), can now filter log messages through a user-specified program.

- All messages printed by the kernel are now routed to the logical device */dev/klog*, which is read by *syslogd*(1M). Kernel messages are filtered through */usr/adm/klogpp* which translates disk device names into mounted file system names.

- *syslogd*(1M) has been updated to the 4.3BSD version. It now supports the routing of messages to a remote *syslogd*.

- *syslogd*(1M) no longer creates the file */etc/syslog.pid* containing the process id of the *syslogd* process. The *killall*(1M) command provides the ability to send a signal to a process by name, so keeping the process id of *syslogd* around is no longer necessary. Note that this requires changing the entry in the *crontab* for *root* which switches to a new system log file every Sunday morning. In Release 4D1-3.2, this *crontab* entry included the following *kill*(1) command to send a SIGHUP to *syslogd*(1M) to cause it to restart with the new log file:

  ```
  kill  -1  `cat /etc/syslog*.pid`
  ```

  In Release 4D1-3.3, the corresponding portion of the *crontab* entry reads:

  ```
  killall  1  syslogd
  ```

  Release 4D1-3.3 contains a new version of the file

  /usr/spool/cron/crontabs/root

  that incorporates the above change, but this file is considered a configuration file. This means that when Release 4D1-3.3 is installed on a system, the file will not be replaced by the new version unless the pre-existing version has not been modified in any way.

  If you have made any changes to the above file to customize it for your particular needs, then after the installation of Release 4D1-3.3, the new version of the file will exist as */usr/spool/cron/crontabs/root.N*. In that case, incorporate the above change so that *syslogd* will receive the SIGHUP that causes it to restart with the new log file.

- The *finger*(1) command has been enhanced to be like the 4.3BSD version. The Berkeley-dependent phone number format has been removed. *finger* also understands both shorter and longer phone numbers.

- *man*(1) now supports the BSD options **–k** and **–f** for performing keyword searches in the NAME sections of manual pages. The alternate forms for man **–k** and man **–f**, *apropos*(1) and *whatis*(1), are also supported. The NAME section database required to support the manual page keyword searches is created using the new command *makewhatis*(1M).

- The default paging program used by *man*(1) has been changed to

  ```
  ul | more -s -f -k
  ```

  This change permits you to scan back through previously viewed pages of a man page using the scroll bar in a *wsh*(1) window. This change also prevents the viewing of manual pages from clobbering the log of recent commands displayed in a *wsh*(1) window. The old behavior can be achieved by setting the environment variable PAGER or MANPAGER to

  ```
  ul | page -s -f -k
  ```

- As a result of the widespread recognition of the security problems associated with *setuid* and *setgid* shell scripts, support for that feature has been disabled in Release 4D1-3.3 of IRIX. An attempt to execute a shell script with the *setuid* (or *setgid*) permission bit set will result in the error EPERM, unless it happens that the owner of the file (or group of the file if *setgid*) happens to be equal to the effective *uid* (*gid*, respectively) of the executing processexecuting process. In Release 4D1-3.3, it is possible to re-enable the support for *setuid* and *setgid* shell scripts by reconfiguring the kernel, but this configuration mechanism is provided only to ease the transition and will likely be removed in the next major software release. If you want to allow *setuid* shell scripts on Release 4D1-3.3 IRIX, edit the file */usr/sysgen/master.d/kernel* and change the value for the variable `nosuidshells` to zero. The relevant line is shipped as:

  ```
  int nosuidshells = 1;
  ```

  meaning that *setuid* shell scripts are not allowed by the system as it is shipped. After changing the value to be zero, rebuild the kernel using *lboot*(1M).

- The *sar*(1) **–t** option display headers have been changed to be more mnemonic. An **–h** option has been added to display kernel heap statistics.

- The *gr_osview*(1) **bbuf** display has been removed -- buffer cache information is now a part of the **rmem** display. An **rmemc** display has been added which is a superset of **rmem.** It distinguishes free memory which cache file system data from those that don't. **net** and **netif** bars display network activity.

- The *su*(1M) command now looks at the user's *.rhosts* file before prompting for a password. See *su*(1M) for details.

5.2 Changes to Networking

The 4D1-3.3 software release includes these changes to networking:

- Security holes in *rlogind(1M), rshd(1M), rcp(1M), rdist(1C),* and *sendmail(1M),* have been fixed.
- *tftpd(1M)* has been made more secure. It logs remote requests if the –l option is specified in */usr/etc/inetd.conf*. The –s option rejects requests for files specified by an absolute pathname (the default). Also, you can restrict requests to certain directories by specifying the list on *tftpd*'s command line in *inetd.conf.* See *tftpd* for details.
- The *timed*(1M) daemon now automatically adjusts the frequency of the system clock for far more accurate timekeeping.
- The network startup script */etc/init.d/network* now starts *named*(1M) before mounting NFS file systems.
- To specify command-line options for the *inetd* and *rwhod* daemons, create *inetd.options* and *rwhod.options* files in */etc/config*. The network startup script now looks for these files when it starts the daemons. See *network*(1M), *inetd*(1M), and *rwhod*(1M) for details.
- *inetd*(1M) now supports RFC1078's TCP port multiplexer, TCPMUX. It allows you to add services started from *inetd* using an assigned port number. The maximum number of arguments for a server is now 11 (the first argument must be the program name). See the man page for details. See *~4Dgifts/examples/network* for sample TCPMUX client and server programs.
- *ftp*(1C),*ftpd*(1M), *rlogind*(1M), and *rshd*(1M) are now based on the BSD Networking Release.

- The file */etc/issue*, if it exists on your system, is displayed at the start of *rlogin*(1C), *telnet*(1C), and *ftp*(1C) sessions.
- Remote logins via *rlogin*(1C) and *telnet*(1C) are disabled if the file */etc/nologin* exists. The file can contain a message, which will be printed before the session is terminated. Remote shell *rsh*(1C) accesses are also disabled but the file's contents are not displayed.
- *login*(1) can now log remote login attempts to *syslogd*(1M). See the *login*(1) manual entry for details.
- *ifconfig*(1M) now prints the netmask with a leading 0x, which is the same format expected by the netmask command line parameter.
- *telnet*(1C) now negotiates the following options with the remote telnet server at startup: window size, terminal speed, local flow control, and line mode. *telnetd*(1M) supports these new options except for line mode.
- *ftp*(1C) has the following new commands: *chmod*, *idle*, *modtime*, *newer*, *umask*, and *site*.
- *ftp*(1M) now supports "restricted" accounts. A restricted account has access restricted to files in its home directory — it is basically an "anonymous" account with a password.
- *rshd*(1M) now requires the account to have a local directory. The **–n** option disables keep-alives. The **–l** option disables use of every user's *.rhosts* file.
- Sample versions of *named*(1M) database files are now installed in */usr/etc/named.d/Examples*. Files in */usr/etc/named.d* are not touched by the installation program (except for the README file.) Put your versions of the database files in this directory. The *root.cache* file has been updated — you should update your copy of it.
- Various stream management bugs in *named*(1M) have been fixed.
- The *nslookup*(1C) utility has been enhanced. If you enter an Internet address when the query type is set to "A" (address), *nslookup* will find its corresponding host name. Before, you had to use the PTR query type to convert the address into a name. Internally, *nslookup* uses PTR queries instead of now-obsolete inverse queries for the address-to-name lookup. The **ls –t** *type* command lists data records of the specified type for a domain. The default root server has been changed to *ns.nic.ddn.mil.*

- The */usr/etc/resolv.conf* file lets you specify the ordering of the Yellow Pages, BIND and file-based host-address lookup services used by *gethostbyname*(3N) and *gethostbyaddr*(3N). Also, you can now specify the default domain search list for BIND queries. See the description of *hostresorder* and *search* in the *resolver*(4) manual page.
- *gethostbyname*(3N) has been enhanced to look for host aliases when doing YP, BIND or file-based lookups. In previous releases, only BIND lookups allowed aliases. See *hostname*(5) for details on how to specify aliases. Also, *gethostbyname*(3N) now returns a valid hostent structure for lookup request that are Internet addresses. This means you can specify a host's Internet address to any network command that expects a host name, such as *rcp*(1C) and *rlogin*(1C).
- The Cornell super-routing daemon, *gated*(1M), is now supported. It is in the *eoe1.sw.ipgate* subsystem. *gated* handles the HELLO and EGP routing protocols in addition to RIP.
- IRIX now has full support for IP multicasting and the IGMP protocol. This implementation is based on the Stanford Multicast 1.2 release.

  Note: This RFC-1112 level 2 implementation of IP multicasting/IGMP is experimental and subject to change in order to track future BSD UNIX releases.

— Chapter 3 in the *Network Communications Guide* describes the programming interface. See *~4Dgifts/examples/network* for a sample multicasting program.

— A daemon for routing multicast packets, *mrouted*(1M), is included in the *eoe1.sw.ipgate* subsystem. This subsystem is needed only on machines with more than 1 network interface.

— The *dog*(6D) demo now uses IP/UDP multicasting by default. *rwhod*(1M) can be configured to use multicasting instead of broadcasting.

— See Chapter 7 of these release notes for known problems with multicasting.

- The *ping*(1M) command has been enhanced. You can specify the time-to-live and output interface for pings sent to an IP multicast address. The **–p** option lets you specify the data to be sent. In the previous release, the optional packet size, count and preload arguments were specified after the hostname. Now they must be specified using the **–s**, **–c**, and **–l** options.
- *fingerd*(1M) has a new –S option (secure) to disallow requests to see who is currently logged in.
- UUCP support has been upgraded to System V Release 3.2. Additional modems, such as the Telebit T2500, have been added to the Dialers file.

5.3 Changes to Documentation

- The *TCP/IP User's Guide* has been completely revised for this release and has been renamed to *Network Communications Guide.* It now documents UNIX domain sockets, IP multicasting, and Sun Remote Procedure Call (RPC) programming.

5.3.1 On-line Release Notes

The options to the *relnotes* command for viewing on-line release notes have changed slightly. When you install the on-line documentation for a product, you can view the release notes on your screen as you would an on-line manual page. However, unlike the on-line manual pages, the printed hard copy of these release notes is more up to date than the on-line version.

The *relnotes* command now accepts the following argments:

–h	describes how to use *relnotes*
product	displays a table of contents for a given *product*
product chapter	displays the given *chapter* of *product*
–t *product chapter*	sends a copy of the given *chapter* to the default printer

To see a description of how to use the on-line release notes tool, type:

```
relnotes -h
```

To see which products have on-line release notes installed, type:

```
relnotes
```

To see which chapters of a product are installed, enter the command below, replacing the word *product* with a product name generated when you type the previous command.

`relnotes` *product*

To view a specific chapter of *product*, replace the words *product* and *chapter* in the following command:

`relnotes` *product chapter*

To page through a chapter, press **`<space>`** and to quit, press **`<del>`** or **`<Ctrl-C>`**. See the *relnotes*(1) manual entry for more information.

5.4 Changes to Graphics

This section contains information on changes to the Graphics Library and the Distributed Graphics Library.

5.4.1 Changes to the Graphics Library

- All the `extern` routine declarations in *<gl/gl.h>* are now prototyped. If your code calls Graphics Library routines with the wrong number of arguments, it will no longer compile. Calls to *setcursor* are the most likely offenders, since it is specified as ignoring its final two arguments.

  If you give *cc* the **–prototypes** flag as we recommend, you will get warnings when the type of an actual pointer argument to a routine is different from that specified in its prototype. e.g. `long *` vs.

`unsigned long *`. Casting the argument to the correct type will eliminate the warning. Note that the form of the cast for arguments that are 2-dimensional arrays is `(`*element–type*` (*) [`*n*`])`, e.g. the cast for the *parray* argument to the *poly* routine is `(Coord (*) [3])`.

- A section for obsolete symbols has been added to *<gl/gl.h>*, *<gl/device.h>*, and *<gl/get.h>*. Symbols found in these sections should not be used in new development. In particular, you should convert all uses of the typedef `Cursor` to `unsigned short[16]`; this will prevent conflicts with X11's typedef of the same name.

- The 256 vertex limit for points drawn with *bgnpoint*, *v*, *endpoint* and lines drawn with *bgnline*, *v*, *endline* *bgnclosedline*, *v*, *endclosedline* has been lifted. The limitation still exists for polygons.

- The name space of GL devices has been partitioned as follows:

0x0000 → 0x0FFF	Devices defined by SGI
0x0001 → 0x00FF	Buttons
0x0100 → 0x01FF	Valuators
0x0200 → 0x02FF	Pseudo devices
0x0300 → 0x0EFF	Reserved
0x0F00 → 0x0FFF	Additional buttons
0x1000 → 0x7FFF	Devices defined by users
0x1000 → 0x2FFF	Buttons
0x3000 → 0x3FFF	Valuators
0x4000 → 0x7FFF	Pseudo devices
0x8000 → 0xFFFF	Cannot be used

The definitions of the *COUNT symbols in *<gl/device.h>* have been changed so that the ISBUTTON() and ISVALUATOR() macros evaluate to TRUE for arguments that fall within the ranges specified above for buttons and valuators.

- The types of a number of GL routine arguments have changed to provide a consistent interface where all color, pixels, bitmasks and device arguments are unsigned. The changes are:

Routine	Argument	Old Type	New Type
czclear	cval	long	unsigned long
defpattern	mask	short *	unsigned short *
defrasterfont	raster	short *	unsigned short *
getdev	devs	short *	Device *
linesmooth	mode	long	unsigned long
lrectread	parray	long	unsigned long
lrectwrite	parray	long	unsigned long
pagecolor	pcolor	short	Colorindex
pntsmooth	mode	long	unsigned long
qenter	qtype	short	Device
setdblights	mask	long	unsigned long
setpup	mode	long	unsigned long
zdraw	b	long	Boolean
zwritemask	mask	long	unsigned long

- A large number of new *getgdesc* inquiries have been defined. see *getgdesc*(3G) for more information.

- The z-range mapping controlled by the GLC_ZRANGEMAP mode to *glcompat* now occurs when a display-list is interpreted. In the previous release, it occurred when the display-list was built.

- The symbol CPOSX_INVALID is no longer defined in <*gl/gl.h*>.

- Calling a GL routine on a machine that does not implement it is now always a nop; error messages are now never issued.

- *greset* no longer partially resets the input subsystem. However, it is still reset by *ginit* and *gbegin*. See *greset*(3G).

- *gbegin* now does what its manual page specifies: a *ginit* except for altering the color map.

- *mmode*

  Upon transitioning to MVIEWING or MPROJECTION mode, the ModelView and Projection matricies are both set to the identity matrix. Upon transitioning back to MSINGLE mode, the single matrix is also set to the identity matrix. In both cases, the matrix stack is reset.

- The following headers are no longer part of */usr/include/gl* on the IRIS-4D B and G:

```
file.h
ic.h
obj.h
object.h
winreq.h
```

5.4.2 Changes to the Distributed Graphics Library

The *dglopen* call now looks at the environment variable *DGLBUFSIZE* to obtain a default communications buffer size. If this variable is not defined, a default value of 4K is used. This call is useful over networks whose transmission unit is greater than the 4K default size.

The variable *DGLBUFSIZE* can be specified as a decimal number indicating the number of bytes, as a decimal number immediately followed by a *K* or *k* (indicating kilobytes), or as a decimal number immediately followed by an *M* or *m* (indicating megabytes). For example, to specify a 2 megabyte buffer, *DGLBUFSIZE* can be set to 2097152, 2048K, or 2M.

5.5 Changes to Demos

- *dog*(6D) now sends multicast packets by default instead of broadcast packets. Multicasting eliminates the load on other machines not using *dog*. For dogfights against 3000s and previous IRIS-4D Series releases, use the **–b** option to specify broadcast mode. If the system cannot set up the socket in multicast mode, it uses broadcast mode. The **–t** option specifies the maximum number of times the packet can be forwarded between networks.

- *buttonfly*(6D) buttons have been reorganized and the button files have been moved out of the */usr/demos* directory into a directory structure that mirrors the button hierarchy.

5.6 Changes to Program Development Tools

This section explains changes to the following program development tools:

- C header files
- C compiler
- error messages
- *dbx*
- other tools

5.6.1 Changes to C Headers

Nearly all the files in */usr/include* have been changed to use ANSI C-style function prototypes.

Since the */usr/include* headers have been updated to return **void *** where appropriate (according to ANSI), some practices that used to work no longer work:

■ C Program: illegal fragment

```
#include <string.h>
main()
{
    char *cp; int n;
    cp = memcpy(x,y,5) + n;
}
```

now generates an error, since *memcpy* returns a **void *** and you cannot increment a **void ***.

■ C Program:

```
cp = ((char *)memcpy(x,y,5)) + n;
```

is (normally) the appropriate change.

5.6.2 Changes to the Compilation Environment

The default space allocation for basic blocks in the optimizer was increased from 500 to 1000. Use **–Olimit** to alter the limit.

The maximum length of a string that the compilers will emit in assembler code (see the –S flag to *cc*(1)) has been increased from 256 bytes to 1024 bytes. If your source code results in a line length in assembler that is larger than your chosen viewer can handle, try using *fold*(1) to fold the lines so you can see the assembler code.

5.6.3 Changes to the C Compiler

- The types **void** and **void *** have been implemented.
- The type qualifier **const** has been implemented.
- Function prototypes are much more fully implemented and checked. For example, a cast of a function-pointer to a prototyped function-pointer works correctly. Completion of incomplete types in function prototypes is not done in messy circumstances (where, for example, multiple declarations of one function exist but complete types are found only by unifying all the declarations).
- The new *cc* option, **–prototypes** turns on type checking of function prototypes, and **–noprototypes** turns it off. The checking is off by default so code that compiled with Release 4D1-3.2 should compile without complaint with this release. The option affects checking of function prototypes, not code generation, so programs compiled either way should work identically.

In checking prototypes, the compiler allows various constructs that are not formally equivalent to get by with a warning to allow old code to compile with the */usr/include* files supplied with this release. Differences such as declaring a function as returning a **char *** at one place and a **void *** at another receive a warning message if and only if **–prototypes** is on.

Prototype differences such as declaring an argument as both **unsigned short *** and **short *** receive a warning only if **–prototypes** is on.

The compiler attempts to be specific about the type discrepancies in function prototypes. Wherever possible it reports the types in English. Wherever possible it also reports the argument number.

If you assign one function pointer to another, the function prototype on the right is the function prototype used unless the right function pointer has no prototype, in which case the function prototype, if any, of the pointer on the left is kept.

- The ANSI C convention that adjacent string literals become a single string (ANSI section 3.1.4, December 1988 draft) is implemented in this compiler.
- For future compatibility, do not put text after `#else` or `#endif` unless the text is enclosed in `/*   */` comment delimiters. While this release ignores arbitrary text on such preprocessor lines, ANSI C considers text after `#else` or `#endif` an error unless it is inside a comment.
- The **–acpp** option to *cc* has been added. When **–acpp** is used, an ANSI C preprocessor is used instead of the normal C preprocessor. Source code for **acpp** (copyrighted by the Free Software Foundation) is in 4Dgifts. See the source for terms of the copyright.

Changes to Error Messages

There are many new error messages. Some of them might be confusing. These in particular have proven to be troublesome:

- `"prototype:pointer to struct is different from actual: pointer to struct"`

A prototype and an actual argument must be pointers to the same struct to be compatible. It is not enough that the structs pointed to be the same size.

- "unknown size: void has no size"

  Many operations on pointers imply taking the size of the item pointed at. **void *** does not point at any object, so operations requiring the size are illegal.

  ■ C Program:

```
void *mfunc();
/* illegal:  pointer arithmetic  */
mfunc(x,y,z)+ 3;
/* legal: pointer comparison to 0 */
!mfunc(x,y,z);
```

- "constant evaluation found questionable result"

  The full message is:

```
constant evaluation found questionable result as a
result of copy propagation or other optimizations.
Constant evaluation left to run time for this case
```

  A warning like this can come from the optimizer. If, as a result of constant propagation or other optimizations, a value is computed that produces an overflow, the optimizer does not do the operation at compile time but leaves it to run-time.

  For example, (*m* << *n*) is undefined in C if *n* is greater than *m*'s width in bits. If you take the meaning as ''do the shift'' the result is zero! If the optimizer discovers that n is greater than 31 (for *m*, a constant int whose value is known to the optimizer) as a result of copy propagation, the shift is left to run-time and the message above is emitted.

5.6.4 Changes to dbx

- C enumerated types are printed much more fully. To see an example of this, try `whatis x` on some enumerated type x.
- `whatis` tries to report on files and variables when they have the same name. If `x` is a file named x.c and a global variable x, `whatis x` reports both usages.
- As of this release, the kernel (IRIX) marks any file being *dbx*'d as busy. Consequently the old practice of doing a recompile and relink followed by the *dbx* **rerun** command does not work unless you first *rm*(1) the executable, because *ld* is unable to write to the busy text file. The lack of a busy mark in previous releases was quite dangerous, because you could relink and then do a *dbx cont* command on text that had changed. This resulted in serious problems during the debugging session.
- A great deal of information was added to the *dbx* manual page and help file */usr/lib/dbx.help*. The information in these files might be helpful to you.
- The *ccall* command can now handle variables in interactive calls to functions in the program being debugged. Interactive calls can now use variables as arguments.
- The **$nextbreak** debugger variable now controls how the **next** command is executed. By default, **next** is done at full speed. If done in the middle of recursions, **next** stops the next time the function is recursively called. If that is not what is desired, set **$nextbreak** to 0 or 1 before issuing **next**. See the help file for further details.

5.6.5 Changes to Other Tools

- *lint*, *cxref*, *ctags*, *cb*, and *cflow* understand the new ANSI C keywords— `const, volatile, signed`— as well as function prototypes. These now work properly on any code that compiles with *cc*.
- *lint* now checks `printf` and `scanf` calls for accuracy.
- Bodyless `if` and `else` clauses are diagnosed by *lint*.
- *lint* mentions possible precedence confusions in many more circumstances than in previous releases.

- *lint* has the *cc* **–prototypes** flag turned on. It is advisable to compile code with **cc –prototypes** before running *lint.*

- *ld* now accepts the –U option. –U makes references to undefined symbol warnings instead of errors. This is useful in certain very large programs where leaving out the code for certain functions saves significant link time and where, for debugging purposes, those functions are not executed. Attempting to execute a missing function results in a segmentation violation.

- Several changes have been made to *make*(1):

— The new command line option **–u** has been added that causes all specified targets to be built unconditionally.

— The *sinclude* directive has been added. *sinclude* works the same way that *include* does in a *makefile*, except that if the file specified in the *sinclude* command is not readable, the error is silently ignored by *make*(1).

— *make*(1) now understands a convention similar to the alternate interpreter feature of *exec*(2). If the first line of a make command file starts with a string of the form "#!alternate_make", then *make* will attempt to invoke the specified alternate *make* with the same environment and arguments.

— The function of the VPATH macro is now documented in the *make*(1) manual entry.

— The behavior of VPATH has been enhanced to allow dependency rules to reference either the virtual target or the actual target. For example, either of the following may be specified:

```
foo.o:foo.h
$(VPATH)/foo.o:foo.h
```

— The interaction between dependency rules and single suffix rules has been corrected. In previous versions of *make*(1), the presence of a dependency rule of the form

```
cat:stdio.h
```

caused the single suffix rule for .c: to be ignored. In the Release 4D1-3.3 version of *make*, single suffix rules are no longer ignored in this case.

— The sizes of many of the internal tables used by *make*(1) have been increased or made dynamic to support long file names and very complex make command files.

- Several changes have been made to *pmake*(1):

— *.f* (Fortran) files were added to RCS transformation rules.

— the new **–u** option (''unconditional'') remakes the specified targets (just like *make*(1)).

— *pmake* now understands the *sinclude* directive (just like *make*(1)).

— *pmake* now sets the MAKEFILE environment variable. It also puts command-line variables into the environment.

— several makefiles can be specified on the command line with multiple –f options.

— the new special targets *.NOTPARALLEL*, *.ORDER*, and *.SINGLESHELL* have been added and are described in the manual page.

— *pmake* does not support rules for checking out SCCS files. It also ignores *make*(1)'s MAKEFLAGS environment variable.

5.7 Changes to the Software Installation Tools

The following are the major changes to the software installation tools for the 4D1-3.3 release:

5.7.1 Changes to Inst

- The Manual Installation menu *defaults* item has been changed to *standard*. The functionality has not changed.

- The *recalculate* item has been added to the `Manual Installation` and `Subsystem Selection` menus. It serves to initiate recomputation of the disk space requirements. A warning is given in this regard after shell escapes.
- The `Subsystem Selection` menu has been reordered.
- The *yes* item of the Selection menu has been changed to *install*, and the *no* item has been split into *keep*, for no action, and *remove*, to remove any old version that might be present. The *remove* requests are now queued along with *install* requests, rather than being executed immediately.
- Removal requests may be made for installed products that are not part of the current distribution listing. These requests will be displayed by the *list* item.
- The *install* item of the Selection menu has been changed to *go*, which serves to initiate the queued installation and removal requests.
- The *versions* menu item serves to directly invoke the functionality of the *versions* command from within *inst*. The *versions* command may still be invoked as a separate IRIX command.
- The *sh* and *shroot* items may no longer be invoked from the `Interrupt/Error` menu.
- Many error messages have changed.
- There is a dramatic increase in the speed of certain network operations.
- The reliability and robustness of the on-line installation history updating mechanisms have been improved. There is no longer a separate *recovery* operation; retrying a failed installation or removal will usually correct the history.
- There is now a check for incompatible subsystems being installed on the disk. The installation tool will not permit you to *quit* until this problem is resolved.

5.7.2 Changes to Versions

- *Versions* is now built into *inst*, and can be invoked as a menu item at any time.
- New products are being produced with dates where old ones had internal version numbers. The –n option can be used to display version numbers in the old form.
- The –q option prevents all output, returning only exit status.
- The format of the display has been improved.

5.8 Changes to 4Dgifts

As in previous system software releases, Release 4D1-3.3 includes a set of on-line source code examples on the DEV (development) tape in the *dev.sw.giftssrc* subsystem. These gifts are not installed or updated by default. To use them, you must first manually install them using *inst*(1M). For detailed information about using *inst* and the manual installation features, see Chapter 2, ''Installing Software''.

Once the gifts are installed, you will find these source code examples in a sample user account directory called */usr/people/4Dgifts*. *4Dgifts* is set up as a sample user account to allow you to learn by example and to allow you to be productive while you are learning to use and customize the NeWS environment.

You should find a *README* file in virtually every directory in and including *~4Dgifts*. The *README* files describe the contents of each directory and provides information that will help you locate the types of source examples you are looking for. Read the file */usr/people/4Dgifts/README* to for an explanation complete understanding of the contents of 4Dgifts.

The source examples provided include programs demonstrating:

- Fortran and C graphics
- the fontmanager

- how to use peripherals (e.g., the dial and button box)
- the Hitachi digitizer (tablet)
- a simple generic SCSI device driver
- some cps, network, UNIX, and video examples
- the January 1989 version of Kermit from Columbia University (This is provided only as a courtesy — Silicon Graphics does **not** support Kermit. See the *README* file in the kermit directory.)
- a complete set of Silicon Graphics image libraries for creation and manipulation of image files
- programs demonstrating a sample WorkSpace environment

5.8.1 New 4Dgifts Directories

New directories of note are:

- *~4Dgifts/examples/audio*

  An audio program for 4D/20 and 4D/25 Personal Iris machines demonstrating how to digitally record microphone input, place the data in a file, as well as playing back the sounds recorded.

- *~4Dgifts/examples/glpg*

  This subtree contains on-line versions of the source code examples given in the *Graphics Library Programming Guide*, Version 2.0 (with the exception of Chapter 18).

- *~4Dgifts/examples/network*

  This directory replaces the previous incarnation, *tcp*, and demonstrates how to use 4.3BSD Internet and UNIX domain sockets on IRIX.

- *~4Dgifts/src/acpp*

  Contains a set of files that comprise the ANSI C compatible pre-processor accessible via the *cc*(1) flag **–acpp**.

- *~4Dgifts/src/dglfax*

  Contains source code for the *dglfax*(1) utility.

- *~4Dgifts/src/sphere*

  Contains source code for the molecule demo */usr/demos/bin/mview*, which makes use of *libsphere.a*, to show how functions in the sphere library are used.

- *~4Dgifts/tutorials*

  Contains interactive graphics tutorials previously only available to those who have attended SGI's 4D Graphics and Advanced Graphics courses. Areas covered include color, light, basic drawing primitives, and modeling and projection transformations. All tutorials are implemented with a user interface that allows you to interactively change the different parameters in real-time and see the effects these changes generate.

- *~4Dgifts/wspace*

  Contains WorkSpace goodies including:

  a subdirectory containing example *.ftr* files and scripts,
  a file containing ''poweruser'' short cut capabilities.

5.8.2 4Dgifts Subdirectory Descriptions

The main subdirectories in *4Dgifts* are: *examples*, *kermit*, *iristools*, *src*, and *tutorials*. The *examples* directory contains various subdirectories housing an assortment of code examples:

Fortran	graphics programs written in Fortran
audio	code demonstrating how to digitally record microphone input to a Personal Iris (only works on 4D/20 and 4D/25 machines), and play back these sounds
cps	four separate subdirectories each with a different type of example implementing the C to PostScript interface
devices	digitizer, dial and button box programs, as well as a program that uses the */dev/scsi* generic SCSI driver
fontmanager	includes sample programs demonstrating usage of the Fontmanager Library, *libfm.a*

glpg	on-line versions of all source code examples contain in Chapter 18 of the *Graphics Library Programming Guide*, Version 2.0.
grafix	various C graphics programs
nurbs	contains four NURBS sample programs: one written in C, one for the DGL, one in Fortran, and one in Pascal
network	has some sample network programs using Internet (TCP/UDP) and UNIX domain sockets, as discussed in Chapter 3 of the *Network Communications Guide*. This directory was called *tcp* in previous releases.
trackball	contains four components of code for a Virtual Trackball Implementation: routines to calculate the virtual trackball, event-queue handling, drive a user-interface, and a simple program to use the other three
unix	contains fundamental examples of system programming
video	contains programs demonstrating usage of various video modes

The *kermit* directory holds the public domain source and documentation for *kermit*, a file transfer protocol that is useful when you need to send files to and from an IRIS/UNIX computer and non-UNIX configurations such as VMS or DOS-based machines.

The *iristools* directory contains a superset of special image libraries, image processing utilities, and graphics tools that used to exist, in the IRIS 3000 Series computers, under */usr/people/gifts/mextools*. This source was used to build the binaries that now reside in */usr/sbin* (e.g., *cedit, showmap, ipaste, mag*). In other words, every executable in */usr/sbin* with a source file under *iristools* was built from that exact source (including the two libraries *libgutil.a* and *libimage.a* under *iristools*).

The *src* directory contains an ANSI C preprocessor in "acpp", a "dglfax" subdir containing the source code for *dglfax*(1), and the directory "sphere" which houses the source for mview (the molecule demo) which shows how functions in the new sphere library--libsphere.a--are used.

The *tutorials* subdirectory contains a set of interactive graphics tutorial programs that cover such areas of the GL as color, light, ortho2, projection, basic drawing primitives, and modeling transformations. All of these tutorials are implemented with a user interface that enables one to interactively change the different parameters in real-time and see the effects these changes generate.

The directory */usr/people/4Dgifts* is setup to work as a sample NeWS user login account replete with many template *.ps* files to help you understand the extent to which you can customize the NeWS environment. Along with a more substantial *user.ps* file, there is a *startup.ps* file, as well as a subdirectory */usr/people/4Dgifts/.4sight.* This directory contains nine additional *startup.ps* files that show you how to create your own user-defined icons, window colors, menu fonts, et cetera. These files have comments throughout them to help describe what they do. There are many ways you can change and alter all of the possible startup variables.

Special Gifts

./iristools/imgtools/snapshot.c

This program allows you to interactively grab part or all of an image on the screen and dump it into an image file. It is the next generation of *icut.* By default, it is loaded in */usr/sbin* as a *gltool.* See *snapshot*(6D). This image file can then be put back up on the screen with *ipaste*(1G), or sent to a supported printer with *lp*(1).

./{.workspace/, README.wspace}*

There is an initial setup for a version of *workspace* with *4Dgifts* that resides in the directory *.workspace.* The file *README.wspace* describes more of what is currently included.

./examples/grafix/{zrgb.c, zrgbmenu.c, zcmapmenu.c}

./examples/Fortran/{zrgb.f, zrgbmenu.f, zcmapmenu.f}

These programs demonstrate aspects of z-buffering in various implementations. Of particular note are the *zcmapmenu* versions, which include a powerful example in the main infinite loop of how to write code that does not eat up extra CPU cycles (provided you do not need the animation to continue when the input focus is elsewhere)

./examples/devices/{iisc.c, inquire.c}

Two programs that use the */dev/scsi* generic SCSI driver. Be sure to also consult the *README* file in this directory.

./examples/audio/audio.c

This program demonstrates how to digitally record microphone input on a Personal Iris (4D/20 or 4D/25) into a file, as well as playing back the sounds already recorded.

./examples/cps/{TagTestc, envdesign, item, vinews} These four directories contain examples that use the C to PostScript (*cps*) interface.

5.8.3 Installing the Gifts

To install the *4Dgifts*, log in as *root*, type **inst**, and follow the instructions on your screen. Refer to Chapter 2, ''Installing Software'', for a detailed discussion of *inst*. Choose the manual installation features, and explicitly select subsystem *dev.sw.giftssrc*. Once you have specified that you wish to use the manual installation features, type:

```
select
```

from the "Manual>" menu and then enter

```
install dev.sw.giftssrc
```

Now 4Dgifts will be included when you run the *go* menu item. The size of this account (uncompiled) is approximately 7133 blocks or about 3.56MB.

5.8.4 Setting Up 4Dgifts as a User Login Account

Upon successful completion of loading the *dev.sw.giftssrc* subsystem from the Development tape (see below), you need to perform one more modification in order to set up */usr/people/4Dgifts* as its own account:

1. Log in as *root.*
2. Edit the file */etc/passwd.*

Duplicate the ''guest'' passwd line.

Change every occurrence of the word ''guest'' on this duplicate line to ''4Dgifts''.

3. Write the changes and exit the editor.
4. Now log out of the console screen entirely and log in as *4Dgifts*.

   Startup is different from when you log in as *guest, root* or any of the other ''default'' login accounts. The intent here is that you copy *~4Dgifts/{.4sight, user.ps, startup.ps}* into your home directory and try changing whatever parts you wish to make it place and define things more in the way you prefer.

5.9 Changes to WorkSpace

The 4D1-3.3 release includes these changes to the IRIS WorkSpace:

- Kernel change

  imon no longer blocks on select. Any changes made on the host machine appear immediately, rather than waiting one to three seconds.

- *fam*

  When *fam* is invoked from *inetd.conf*, it can be passed an argument which determines the polling period when it is watching files over NFS. After the new argument is put in *inetd.conf*, you can activate it by becoming *root* and entering **`killall -HUP inetd`**.

  The programatic interface to *fam* was changed so that any *fam* request must be accompanied by the uid and gid of the calling process. *fam* then examines the file system with those settings, reverting to root privelege when it is finished.

  fam correctly reports when a directory disappears.

- Sticky directories

  WorkSpace now deals with the problem of sticky directories. (These are directories which, if they are owned by another user, do not allow you to remove files owned by other users, and on which you don't have write permission, even though you have write permission in that directory).

  Note these behaviors:

  - If no sticky directories are involved, file transactions are generally as they used to be, although they should be much faster for large directories moved around on the same disk.
  - If a sticky directory owned by someone else exists and contains another user's file, and you do not have write permission for that file,

    1. You cannot remove that file.
    2. You cannot drag that file to a new directory.
    3. You cannot remove its parent directory (even if you have full write permission on it).
    4. If you have write permission on the parent, you can drag it to another directory on the same disk, but you cannot drag it across disks.

    These behaviors apply even if the files were dragged to the WorkSpace.

- NFS

  If files are exported without root permissions, root explicitly cannot read them from another machine, unless there is read permission for others. Even if you own a directory on another machine, you cannot mount that directory and list its contents if you are running as root. *fam* now runs with the uid and gid of the calling process, so that you can run WorkSpace, visit remote mounted directories that you own, and view their contents.

The 4D1-3.3 release includes this enhancement to IRIS WorkSpace performance: WorkSpace now updates itself only when all of the *fam* messages for a given burst are read off the select. This speeds up the opening of large windows.

The 4D1-3.3 release includes these improvements to IRIS WorkSpace usability:

- Name length of dumpster is no longer limited. You can rename the dumpster via the prefsheet, with a long name.
- If directory view windows are stowed when WorkSpace is shut down, they now come up stowed when WorkSpace is restarted. Note that the blue WorkSpace window should, however, never come up stowed.
- A stowed directoryView is no longer painted after update events until it is opened, so that however many update events occur, it paints only when it is opened.)
- Arrange as group on WorkSpace should be a no-op if nothing is selected.
- If you drag a file to a directory, and it overwrites the existing one, the new file now appears where you drag it to, instead of overwriting the existing one.
- If *dirview* is requested to open a directory and is unable to do so, it now puts up a notifier naming the file and stating that it can't be opened.
- When you try to do a file transaction and fail, the resultant notifier informs you of the name of the file it failed on.

The 4D1-3.3 release includes these visual improvements to WorkSpace:

- WorkSpace no longer leaves little white lines where the text view was if you move out of a scrolled view.
- The text view no longer repaints itself briefly in the wrong place if you switch from view as list to view as icons or vice versa.
- When an icon is dragged, the icon and the text field now paint themselves at the same time, instead of letting the text field paint first.
- When you run as root for the first time (without a *.workspace* directory), the WorkSpace window is now layed out in a more attractive fashion.
- When you tried to copy a file that was running, WorkSpace formerly created a file on the WorkSpace window, showing the icon as running. Now the new file is shown in its proper state.
- WorkSpace now allows you to make a linked copy of the root folder.
- When you are running with template and ask to restore root, it now gets its connection line immediately, instead of waiting until you move it.

- *getinfo* windows now have **no close** on the menu.
- If you run as root, the name of your dumpster is now /dumpster, not //dumpster.
- When a window with cached data appeared, some icons formerly appeared before others. Now they all appear together.

5.10 Changes to 4Sight

The 4D1-3.3 release includes this change to 4Sight:

- *4Sight* was changed so that whenever a GL window is exited, the cursor color is reset to red. This was done so that programs that modify the cursor color won't interfere with other programs that work better with the default red cursor.

  However, the implication is that if a program does change the cursor color, it must do slightly more than what it did in the release 4D1-3.2. Now, to change the cursor color, a GL program must do the setting of the cursor color every time it receives an ''enter'' INPUTCHANGE event.

6. Bug Fixes

This chapter describes bug fixes to:

- IRIX
- program development tools
- software installation tools
- networking
- the Distributed Graphics Library (DGL)
- mail programs
- demos
- 4Dgifts
- graphics
- 4Sight
- IRIS WorkSpace
- the X Window System

A Silicon Graphics software change request (''SCR'') number appears after many of the bug fixes in this chapter.

6.1 Bug Fixes to IRIX

- Missing arguments to a debugging *printf()* and return statements in *random*(3) have been added.

- The *lpshut*(1M) command would occasionally fail to completely shut down the lp system when the system was printing via the network. *lpshut* has been fixed to ensure that the lp system is actually stopped before it finishes. (SCR 7628)
- The new *syslogd* daemon is provided to log kernel messages (e.g. disk errors) to the disk. (SCR 3079)
- The */etc/shutdown* continuation query message has been clarified. (SCR 3118)
- *vi* tags now generate search forward commands, so the initial tag search will work. Moving below the match and issuing a `:ta` command again will fail (assuming there are no more matches). (SCR 3265)
- The */usr/people/guest/.profile* and */.profile* files no longer set the *$PATH*. The *ls* function and *tset* invocation in these profiles is refined. (SCR 3279)
- The *fs*(4) manual entry now consistently uses the term "*fs*", not "*efs*". (SCR 3319)
- The *continue* statement in *awk* no longer occasionally invokes the *next* statement. (SCR 3493)
- The *join*(1) command no longer core dumps if the number of fields in the input line is less than the *join* field specified by **–j**. (SCR 3604)
- The **–prune** option has been added to the *find*(1) command. (SCR 3790)
- *awk*(1) now has an improved usage error message. (SCR 3925)
- *mv*(1) now maintains file ownership and times when moving files across file systems. (SCR 3941)
- The graphics input queue now allows dynamic size allocation via *lboot*(1M). (SCR 3942)
- *exec*(2) now allows tuning of NCARGS argument list size. (SCR 4145)
- Several dozen words have been added to the *spell* dictionary of known words. (SCR 4364)
- The *bc*(1) command no longer dumps core on the *di* command. (SCR 4502)

- *dircmp* and *file* now recognize dangling symlinks. (SCR 4699)
- *odump* is now documented in the Programmer's Guide as an object file dump command. (SCR 4718)
- *<signal.h>* now incorporates portions of the now obsolete *<bsd/signal.h>*. (SCR 4876)
- *<sys/param.h>* now incorporates portions of the now obsolete *<bsd/sys/param.h>*. (SCR 4883)
- The *nawk* implementation limits have been documented in the *Programmer's Guide*. (SCR 5014)
- Chapter 13, ''Shared Libraries'' of the Programmer's Guide has been rewritten to describe Silicon Graphics's version of the shared libraries (rather than AT&T's version). (SCR 5015)
- The *PATH* environment variable is no longer set in any of the default provided shell scripts (such as various *.profile*, *.cshrc*, and *.login* scripts). A default *PATH* is set by each program that establishes a login session, such as */bin/login*, */etc/gl/pandora*. (SCR 5071)
- Configurable security features have been added to *login*. Logging of successful and unsuccessful login attempts can be enabled with the *syslog* keyword in the configuration file */etc/config/login.options*. The number of attempts permitted and the disable time after unsuccessful attempts can also be configured in */etc/config/login.options*. (SCR 5073)
- The *ls* –F option and special symbols /, * and @ are now documented in the *ls*(1) manual entry. (SCR 5074)
- *tic* now allows longer filenames. (SCR 5232)
- *vi* now handles SIGWINCH to change window size. (SCR 5324)
- The *whatis*(1) BSD command is provided in Release 4D1-3.3. (SCR 5370)
- The *w*(1) BSD command is provided in Release 4D1-3.3. (SCR 5398)
- *<file.h>* no longer requires *<types.h>* to be explicitly included first. (SCR 5534)
- The *expr*(1) manual entry now documents the *match* operator. (SCR 5682)

- The explanation of the *getitimer*(2) return value has been clarified in the manual entry. (SCR 5726)
- *setvaluator* now always generates events properly. (SCR 5745)
- The *hinv* command reports CPU and FPU revisions, replacing the obsolete *showconfig* kernel routine. (SCR 5755)
- *prctl* has been enhanced to allow the sending of a signal to an entire share group upon exit of any member of the group. This allows *sproc* child processes to be signaled if a parent process dies. (SCR 5759)
- The *sadp* command has been deleted since it cannot work with intelligent disk controllers. (SCR 5802)
- The */etc/killall* executable is marked `set group id == sys` to allow it to access */dev/kmem* on behalf of any user. (SCR 5808)
- The IP4 status LEDs now show 1 second "heartbeats". (SCR 5810)
- The argument type description in the *signal*(2) manual entry has been corrected. (SCR 5908)
- The descriptions of the *h* and *l* character positioning commands have been corrected in the *vi*(1) manual entry. (SCR 5947)
- The information on FILES in the *math*(5) manual entry has been corrected. (SCR 6015)
- *pty*: adjusted STREAMS pseudo-terminal driver to avoid resource deadlocks. (SCR 6023)
- The *fclose*(3S) library routine now avoids forcing *errno* to ENOENT. (SCR 6036)
- *<errno.h>* now declares *sys_errlist*[] and *sys_nerr*. (SCR 6054)
- *tar* now provides for partial ability to handle multiple volume backups onto QIC-02 drives. (SCR 6090)
- *ls*(1) manual entry now documents the –L and –H options. (SCR 6129)
- *find*(1): The command `find . -name '+*' -print` now finds those files whose name begins with '+', instead of finding all files. (SCR 6174)
- The */tmp* directory is no longer removed during the boot process if it is a symbolic link or mount point. (SCR 6210)

- *tar*(1) has been fixed so that "tar t <filename-list>" lists only requested filenames. In previous releases, this command listed all names in the tar archive. (SCR 6264)

- The *rint*(3M) math routine is provided to round in direction of rounding mode. (SCR 6370)

- *script*(1) now passes control-Z and control-C to its subshells. (SCR 6398)

- *tail*(1) now buffers up to 256 kilobytes of data. (SCR 6404)

- The *sh*(1) internal command *ulimit* has been fixed to display a value of zero (0) correctly. (SCR 6408)

- *memory*: Release 4D1-3.3 provides improved management of process memory and core file needs. The default core file size is now adjustable. Also, there are no more `out of swap` messages printed. You can adjust the maximum RSS size of a process (thus guaranteeing that some resources are always available) and the maximum stack size.

  The allocation of swap space is now delayed until it is actually needed. This allows a process to *malloc* very large, infrequently-used data areas to have a virtual memory size that is larger than available swap space, as long as the pages of memory that are actually touched fit in swap and memory.

  When swap finally runs out, the ''greediest'' process is selected by the kernel and killed, allowing other processes to continue correctly. (SCR 6416)

- The new command *ident*(1) searches files for RCS keywords. (SCR 6480)

- The curses library *libcurses.a* has been converted to use *ioctl*(FIONREAD) and *select*(2) to improve handling of input. (SCR 6499)

- The BSD system calls *getrlimit*(2) and *setrlimit*(2) have been added to IRIX in this release. The *csh*(1) builtin commands for manipulating resource limits are also provided. (SCR 6515)

- Obsolete kernel *fastcallout* table removed from */usr/sysgen/master.d/kernel*. (SCR 6539)

- The *killall*(1M) manual entry describes handling of numeric process names. (SCR 6541)

- The reference to *mntent.h* in the *setmntent*(3) manual entry is now correct. (SCR 6543)
- *awk* and *oawk* now emit correct line numbers in error messages for *awk* scripts with comments. (SCR 6549)
- The *mknod*(1M) manual entry now identifies the correct location of list of major device numbers. (SCR 6573)
- *vi*(1) no longer creates temporary files that are several times the size of the file being edited. *vi*(1) also now gives more meaningful error message when the system is out of disk space. (SCR 6584)
- Directory access times are now changed if the directory is read, for example, by the *ls*(1) command or the *scandir*(3C) library routine. (SCR 6615)
- ps(1): The output of the -l option now provides a two byte `F` field and wider `SZ` and `TIME` fields. (SCR 6633)
- The RSS output field from the –l option of *ps*(1) is now based on kernel data that is updated every second and the calculation of this value is improved to handle shared regions and mapped physical device memory, as described in the *ps*(1) manual entry. (SCR 6663)
- Declarations for the routines *utmpname*, *setutent* and *endutent* have been added to the include file <*utmp.h*>. (SCR 6667)
- *vi*(1) no longer creates zombie processes when using the `!` command to filter a file's contents. (SCR 6727)
- The kernel's ability to create a crash file in */usr/adm/crash* in the unfortunate event of a panic is more reliable. (SCR 6733)
- The *Mail*(1) command now accurately detects the size of the window it is invoked in. (SCR 6745)
- *stdio.h*: irrelevant portions for VAX and MIPS hardware have been removed. (SCR 6750)
- The *gr_osview* -N option now works with remote host accounts that have password protection, as long as the related *.rhost* file allows access. (SCR 6814)
- The new *setrlimit*(2) system call provides a way to limit core file sizes. (SCR 6887)

- The **-M** and **-d** options to the RCS command *co* have been fixed to calculate time correctly. Also, a possible core dump condition is eliminated. (SCR 6909)

- The *dialwarp* command is now documented in a manual entry. (SCR 6974)

- *setitimer* now allows for better timer resolution (less than 1 millesecond) using *ftimer*. (SCR 6976)

- *at*(1) now uses the correct group ID (gid) to run a task in cases where the user previously changed groups using *newgrp*(1). (SCR 6993)

- The *terminfo* item `sgr0` is now specified in the *terminfo* file for `iris-ansi` TERM type. This allows the request `attrset(A_NORMAL)` to work inside programs, even when following an `attrset(A_BOLD) request.` (SCR 7006)

- The Berkeley *BSDsetpgrp*(2) and *vhangup*(2) routines now have manual entries. (SCR 7011)

- *tpsc*(7M): The tape driver is enhanced to allow writing data after the EOT marker, as required by ANSI X3.27 standards. (SCR 7054)

- */usr/include/gl/dials.h*: Removed bogus comment. (SCR 7073)

- memory allocation: Routines that allocate stack space, such as the GNU routine *alloca*, can now pass the address of that space to the kernel, and expect that the stack will be grown appropriately if needed to create the space.

  Previously, system calls requiring such space could fail with errors such as EFAULT, after the kernel had failed to automatically grow the required space. (SCR 7080)

- *make*(1): Added a description of the `$(@D)` and `$(@F)` macros to the *make*(1) manual entry. (SCR 7106)

- *make*(1): The macro `$(@F: .o= .c)` now works inside a suffix rule. (SCR 7145)

- *m_fork*(3P) no longer produces a segmentation violation. (SCR 7159)

- References to the obsolete *initstate* `5` have been deleted from the *shutdown*(1M) manual entry. (SCR 7174)

- *bstream*(1): The undocumented **-B** option is no longer available. A bug that could cause *bstream* to not write a final, partial sized block to tape has been fixed. A bug that could cause *bstream* to hang has been fixed. (SCR 7301)
- The *make*(1) manual entry now documents VPATH. (SCR 7334)
- *console baud rate*: the *dbaud* parameter in the *bootrom* command monitor now changes the graphics console baud rate. (SCR 7340)
- The *dbg*(4) manual entry is now also available using the name *debug*, for convenience of access. (SCR 7363)
- virtual memory: The performance of the virtual memory paging daemon is improved to minimize time spent computing page reference counts. (SCR 7367)
- The *awk*(1) command now queries the kernel to determine the available number of open file descriptors, instead of allowing just 15 open files. (SCR 7391)
- mkdir(1): The **-p** option now exits with a zero (`0`) status if it is successful. (SCR 7394)
- The *usconfig*(3P) routine now returns the documented values. (SCR 7407)
- The *mkfile*(1M) command is now described in a manual entry. (SCR 7408)
- *syslogd*(1M): The obsolete **-m** option to *syslogd* is no longer mentioned in the manual entry. (SCR 7458)
- The System V semaphore and message IPC structures are now initialized to zeros before being allowed to be accessed from applications with *semget*() or *msgget*(). (SCR 7459)
- The include file */usr/include/sys/invent.h* is now C++ compatible. (SCR 7466)
- The *imon*(7M) command is now documented in a manual entry. (SCR 7467)
- The *halt*(1M) manual entry has been rewritten for improved clarity. (SCR 7488)

- Various virtual memory bugs allowing system hangs or instability in the virtual memory paging system have been fixed. (SCR 7490)
- The *bootp*(1M) manual entry no longer describes the obsolete **-h** option. (SCR 7514)
- The *setsym*(1M) command is now described in a manual entry. (SCR 7574)
- Password aging is now described in the *passwd*(4) manual entry. (SCR 7613)
- *sed*(1) no longer dumps core on an empty expression, such as `'s///'`. (SCR 7637)
- File creation performance has been improved. Creation still writes to the disk synchronously to increase file system robustness, but it doesn't take as long. (SCR 7640)
- The *vi*(1) editor now recognizes changes in the size of the window it is invoked in. (SCR 7646)
- A bug in *regcmp*(3X) that could cause a core dump has been fixed. This bug could be seen by invoking the command `man *`. (SCR 7700)
- *swap*(1M): Fixed various bugs that could cause **`swap -d`** (delete swap space) to fail or to hang the system. (SCR 7704)
- *libsun.a*: The *strdup*() routine, which is available in *libc.a*2, has been removed from *libsun.a*. (SCR 7729)
- The *exit*(2) manual entry now correctly describes *exit*'s effect on open files and its interactions with *sproc*(2). (SCR 7747)
- The *nawk*(1) command no longer core dumps on null bodied functions. (SCR 7772)
- The file */usr/include/dirent.h* now correctly defines MAXNAMLEN to be `255`. (SCR 7775)
- A bug in *gr_osview*(1) that could cause *gr_osview* to hang the system has been fixed. (SCR 7782)
- The following routines have been added to *libds.a*: *read08*(), *reservunit16*() *releaseunit17*(), These routines were already documented in the *dslib*(3) manual entry. (SCR 7800)

- The *pcreate*(3C) routine is now cross-referenced in the manual entries for *fork*(2), *exec*(2), *prctl*(2), *sproc*(2) and *system*(3C). (SCR 7824)
- The descriptions of the *hinv*(1M) options have been corrected in the manual entry. (SCR 7826)
- The *vi*(1) command `<Ctrl>B` now scrolls backward two lines less than a full screen, instead of two lines more. (SCR 7830)
- The directory */usr/bin/X11* has been added to the search path of *whereis*(1). (SCR 7842)
- The default PATH for the *tutor* login has been fixed. (SCR 7852)
- Bugs in the *more*(1) command *(/usr/bsd/more) that could cause more* to leave the terminal in a raw state after an interrupt have been fixed. (SCR 7854)
- The *logname*(1) manual entry has been changed to correctly describe its use of *cuserid*(3S). (SCR 7875)
- The *login*(1) command has been changed to allow for $HOME directory pathnames that are up to PATH_MAX (which is 1024) characters long, instead of only 64 characters. (SCR 7901)
- *savecore*(1): System crash files in located in */usr/adm/crash* are now created with file modes of `0600`. This close as security loophole. (SCR 7904)
- The *sar* **-b** option now checks for zero and negative results to avoid displaying nonsensical values. (SCR 7906)
- The *man*(1) manual entry now documents the PAGER option. (SCR 7925)
- *exec*(2) no longer fails with the error message

  ```
  Insufficient memory to allocate 8 pages
  ```

  when additional free swap space is still available but main memory is all in use. (SCR 7936)
- The *subj*(1) command now works when invoked from *csh*. (SCR 7951)
- The *inittab*(4) manual entry now states that the ID field in */etc/inittab* can have up to four characters. (SCR 7956)

- The *sar* (1) manual entry now describes all of the columns output by the -t option. (SCR 7974)
- The *csh*(1) manual entry now documents the *child* environment variable. (SCR 7975)
- The SHACCT shell script accounting feature is now available for *sh*(1). (SCR 7981)
- The *mt*(1) manual entry now describes where to find the values for the `Status` field output by the *mt status* command. (SCR 8006)
- *more*(1) - Reliably cleans up the `--More--` prompt at completion. (SCR 8018)
- getty(1M) - The getty manual entry documents the –s option, to allow to getty's on the same line to avoid colliding. (SCR 8025)
- mknod(1M) - The mknod manual entry now refers to the correct file for major device number assignment. (SCR 8031)
- cron(1) - The cron command now sets the USER environment variable when running commands. (SCR 8032)
- intro(2) - The introduction to section two of the manual now describes EAGAIN, O_NDELAY, empty streams and locked files. (SCR 8050)
- cpio(1) - The cpio manual entry describes the –H and –L options. (SCR 8100)
- ctags(1) - The ctags manual entry describes the –v option. (SCR 8126)
- sysfs(2) - The sysfs manual entry describes its expected fsname argument. (SCR 8134)
- exec(2) - The exec manual entry now refers to ARG_MAX instead of to the obsolete explicit value of 5120. The current value of ARG_MAX is 10240. (SCR 8168)
- which(1) - The –f option is provided, to suppress reading of /.cshrc. (SCR 8201)
- *apropos*(1) - The BSD Unix commands *apropos* and *makewhatis* are provided. (SCR 8312)
- system(3S) - Thanks to improvements in the virtual memory management of IRIX, the *system*(3S) library routine can now efficiently handle the

invocation of a command from within a large process, without requiring enough swap space to store a temporary additional copy of the large process. (SCR 8366)

- exec(2) - The exec manual entry now discusses share group processes. (SCR 8367)
- vi(1) - The vi editor has improved handling of the numeric keypad. (SCR 8474)
- crontab(1) - The crontab command no longer allows null length files from standard input, to avoid allowing the user to accidentally empty their crontab file. (SCR 8522)
- passwd(4) - The valid values for user id (uid) and group id (id) are extended from 29999 to 60000. (SCR 8597)
- env(1) - The env command no longer has a limit on the number of environmental variables handled. It had been limited to 100 variables. (SCR 8625)
- closedir(3B) - The closedir library routine now works with other malloc and free library routines even when the free routine clears or alters memory while freeing it. Before, the closedir routine could fail to close the file descriptor in use. (SCR 8633)
- more(1) - The more command no longer is confused by requests to skip more pages than remain to be displayed. (SCR 8636)
- login(1) - The login command now supports password aging and limits on the number of successive failed attempts. (SCR 8640)
- diff(1) - The diff command is upgraded to the Berkeley version, and supports the –c context option. (SCR 8653)
- login(1) - Fixed a problem that prevented logging in with graphics when certain other jobs were executing in the background. (SCR 8675)
- sysconf(2) - The number of characters allowed in the exec(2) argument list is now configurable. (SCR 8677)
- acctmerg(1M) - The acctmerg can now merge up to 100 (see getdtablesize(2)) files at once, instead of just 10 files. (SCR 8726)
- shutdown(1M) - The /etc/shutdown script no longer runs /etc/rc0 to terminate multi-user processes, but rather lets init run /etc/rc0 after single

user state is entered. This solves a problem that had prevented entering single user mode while using pandora(1). (SCR 8790)

- rmt(1M) - The remote tape facility is enhanced to provide greater compatibility with other Berkeley Unix based systems and with older IRIX releases. (SCR 8865)
- mousewarp(6D), keywarp(6D) and dialwarp(6D) - These facilities are added to allow user configuration of input device response speeds. (SCR 8866)
- csh(1) - The foreach construct now allows an index variable name that has embedded digits. (SCR 8880)
- sh(1) and csh(1) - The shells now have builtin commands to allow setting limits on coredumpsize (the size of the largest core dump that can be created), memorysize (the maximum amount of physical memory allocated to a single process), and other memory and file limits. By setting a reduced coredumpsize, users can avoid having their system take minutes to dump a large application's core. (SCR 8911)
- spell(1) - The spell command no longer fails in the event that the invoking environment happened to have a variable set by the name of "D". (SCR 9012)
- inittab(4) - The inittab manual entry now describes the (limited) comment conventions of the inittab file. (SCR 9022)
- man(1) - The man command now supports the –k (keyword) option. (SCR 9032)
- multgrps(1) - The Berkeley multiple groups feature is supported. (SCR 9037)
- prtvtoc(1M) - The prtvtoc now displays the correct number of bytes per sector, even on disks formatted by old versions of the fx(1M) command. (SCR 9100)
- documentation - A caveat is included in the Release Notice, providing additional instructions for section 9.2 of the "Writing Device Drivers for Silicon Graphics Computers" guide. These instructions describe how to test a terminal before selecting "setenv console d" in the boot rom monitor. (SCR 9128)
- IRIX - A virtual memory bug in IRIX is fixed, which had caused systems to occassionally hang after many hours of heavy swapping. (SCR 9133)

- su(1M) - The su command now exits with an appropriate message if it is unable to open /dev/tty to obtain a password. (SCR 9137)
- mkdir(2) - Newly created directory blocks are now initialized to zero's. (SCR 9183)
- sh(1) and csh (1): The ability to use setuid and setgid permissions on shell scripts is restricted. This ability is enabled by a kernel configuration option. By default this option is disabled. See the variable nosuidshells in the file /usr/sysgen/master.d/kernel. The setuid shell script /etc/gl/startconsole is rewritten as a C compiled program, so that the setuid shell script feature is no longer required by any Silicon Graphics released software. These changes improve system security. (SCR 9287)
- dcopy - References to this unsupported command are removed from the System Administrator's Guide. (SCR 9340)
- script(1), vi(1) and more(1) - The vi and more commands now work while using the script command to capture terminal sessions. (SCR 9363)

6.2 Bug Fixes to Program Development Tools

6.2.1 C Compiler Bug Fixes

- *cc* now generates a warning when –#, an unsupported option, is used. (SCR 5308)
- The fatal internal error `schain botch` is no longer caused by any source program. (SCR 6106)
- << and >> in constants now get the right signed or unsigned type. (SCR 5681)
- Previously, ++ was ignored in complex circumstances. It is no longer ignored. (SCR 5478)
- `enum` declarations in inner scopes are now handled correctly. (SCR 6145)

- Prototypes are now handled correctly. (SCR 4678, 4895, 5088, 5904, 6153, 6242, 6462)
- Coredumps on illegal programs were fixed. (SCR 4351, 5378, 5458, 7168, 7252)
- *volatile* now generates the correct types and is accepted wherever it is allowed in ANSI C. (SCR 5454, 6378, 6381)
- Floating point expressions (`d > d1`) did not generate an integral `0/1` result. (SCR 7013, 7223)
- *void* and *void** now work. (SCR 3193, 4797, 6010, 6256, 6773, 7285)
- Register declarations sometimes used to cause a *ugen* coredump or error message. (SCR 5263, 5716)
- The ternary operator `?:` with float types used to cause *as1* coredumps. (SCR 6775, 7191, 8484)
- *enum* values are now treated as another name for *int*. (SCR 7136, 8609)
- Some *struct* pointer function prototypes used to cause a *ccom* coredump. (SCR 7321)
- The `const` keyword is now understood. (SCR 6257)
- Structure returns no longer cause argument passing problems. (SCR 8853)
- Using `float to` and `from` unsigned conversions now works correctly. (SCR 5441)
- Several optimizer (*cc –O2*) bugs were fixed. (SCR 4414, 4585, 4624, 4918, 5084, 5351, 5442, 5455, 5577, 6099, 6146, 6371, 7088, 7236, 8581)

6.2.2 C Preprocessor

- Escaped newlines in strings resulted in wrong line numbers being assigned. (SCR 6219)

6.2.3 Other Development Tools

- The assembler would coredump if the user specified an assembly-time divide by zero. (SCR 6981)
- The reorganizer (assembler) would generate illegal reorganizations in certain rare circumstances. (SCR 7332)
- *cflow* now generates type output correctly. (SCR 7283)
- *cxref* had several problems. (SCR 6376, 6380, 6386, 6950, 7357, 8596)
- *ar* now uses the TMPDIR environment variable to determine the location of temporary files. (SCR 5387)
- *ar* used to coredump with certain bad object files as input. (SCR 6041, 6105)
- *nm* used to print out erroneous symbol data under specific circumstances. (SCR 5475)
- *cb* now has improved option checking so that improper options will not silently put it into reading standard input. (SCR 5307)
- *lint* now reports unused variables. (SCR 5374)
- *ldexp*(3C) now returns `0.0` if its first argument is `0.0` instead of treating `0.0` as a denormalized number.
- Several bugs in *make*(1) have been fixed:

— the –p option now doesn't print extraneous things; all output goes to stdout; when –d is on, stderr/stdout are flushed so output isn't interspersed. (SCR 6086)

— the –u option was added to unconditionally execute a command (SCR 6196)

— The manual entry now documents the $(@D) and $(@F) macros (SCR 7106) and VPATH (SCR 7334).

— the macro $(@F: .o= .c) now works inside a suffix rule. (SCR 7145)

- Several bugs in *pmake*(1) have been fixed:

— a missing *rm* was added to the *.sh.out* rule in */usr/include/make/system.mk*

— the incorrect transformation rules for archives were removed

— *pmake* now properly checks lines beginning with `include` properly to decide if the line is a target or include command

6.3 Bug Fixes to Software Installation Tools

- A message will be displayed while the system is coming up if there are configuration files that have changed status during a recent software installation. (SCR 5045)
- Annotation explains the meaning of the `.O` and `.N` suffixes in the *versions config* and *versions changed* output. (SCR 5051)
- The *versions user* command will no longer descend into NFS-mounted directories during its scan of the file system. (SCR 5767)
- The miniroot will now display the current system date and time before the installation tool runs, as a precaution against installing software with an incorrect system date. Resetting, if necessary, can be done through a shell escape. (SCR 5805)
- The installation tool would sometimes retain an incorrect notion of the physical position of the tape when other errors occured. This has been corrected. (SCR 5869)
- The notification of which subsystems cannot be installed on the currently-running IRIX system was not being given at an appropriate time. It has been integrated with the other pre-installation checks. (SCR 5881)
- During deletion of installed products, the tracking of subsystems that had changed names or other grouping was not always done correctly. Subsystem tracking for purposes of removal has been fixed. (SCR 5896)
- The *versions* command has been built into *inst*, and can be run at any time as a menu item, as well as a separate IRIX command. (SCR 5922)
- The default root directory was not being set correctly for *versions* commands executed from a chrooted shell. This caused unexpected *versions* behavior. (SCR 6068)

- Some `disk full` conditions were not properly detected by previous versions of *inst*. Writes to a full disk are now detected, and generate the appropriate error conditions. (SCR 6069, 6070, 7491)
- There were several weaknesses in the file system mounting, unmounting, and making logic in *inst*, sometimes causing incorrect behavior. These have been corrected. (SCR 6252)
- Many reliability issues in the handling of the online installation versions history have been addressed. These improvements solve certain problems where subsystems and files were not being removed properly. (SCR 6253)
- The frequency of disk space recalculations has been greatly reduced. Typically, the disk space calculations are done only once per software distribution access. (SCR 6396, 7554)
- The installation tools now retain a record of all products that are available during the installation process, even those that are not installed. This information tends to improve the value of the default subsystem selections in future installations. (SCR 6410)
- The error message for the `insufficient disk space` condition has been improved. Previously, the name of the offending file system would sometimes not appear. (SCR 6558, 6648, 9025)
- While running in the miniroot, older versions of *inst* would temporarily convert symbolic links to directories from absolute to relative form, so that they would resolve during installation. The path interpretation is now done without altering the link. (SCR 6689, 8128)
- Network timeouts during software installation could sometimes result in a core dump. This has been corrected. (SCR 6695)
- *Inst* no longer attempts to access a remote distribution server as **`root`**; the default is **`guest`**. (SCR 6709)
- The miniroot no longer uses default */etc/sys_id* and */etc/hosts* configuration files. These files are derived at the first attempt to access the network. (SCR 7128)
- The reliability of the installation history updating mechanism has been greatly improved. This will improve the cleanliness of future removals of system software, among other things. The **`recovery`** mechanism is no longer necessary. Normally, repeating a failed operation is sufficient to

correct any problems there may be in the history. Attempts to remove non-empty directories are retried with each new installation or removal until the directory is successfully removed. (SCR 7255, 7415, 7440, 7525)

- The restrictions on installing subsystems under IRIX have been greatly reduced. (SCR 7586, 8241)
- Disk space requirements are now based on a more complete perspective, and are relatively accurate even when the online installation history is incorrect. (SCR 7828)
- The disk space accounting for configuration files is now handled in a more accurate manner. (SCR 7934)
- The miniroot now borrows */usr/etc/resolv.conf*, along with */etc/sys_id* and */etc/hosts*, when configuring the network. (SCR 8244)
- The installation tool is now more flexible about file system configurations, and will never make a new file system without prompting for confirmation. (SCR 8527)

6.4 Bug Fixes to Networking

- Entering a `<Ctrl-D>` on a line with other characters preceding it now no longer causes a logout. (SCR 2960)
- A *ptc* opened with O_NDELAY no longer hangs the program when data is written to the *pty*. (SCR 6033)
- The tty line discipline now behaves like those in other STREAMS implementations. A writer on a slave pty is blocked when it overruns the reader on on the matching controlling pty. Previous releases discarded the excess characters. (SCR 6034)
- */etc/netgroups* can now contain comments. (SCR 6687)
- *ftp* core no longer dumps core when the *open* command is not given an argument. (SCR 7017)

- *tftpd* now is more secure. It is configured by default to restrict access to files in certain directories. All options are documented in the *tftpd*(1M) manual entry. (SCR 7115, 9176, 9046)

- *ruptime*(1C) now reports the load average statistics for the local machine. (SCR 7155)

- The tty driver line discipline was changed to give better response to the interrupt character. (SCR 7238)

- When using *telnet* to access a */bin/sh* account on an IRIS running 4D1-3.2, typing the interrupt character would end the *telnet* session. This has been fixed. (SCR 7319)

- The CMC ENP-10 Ethernet driver now reports late collisions. (SCR 7510)

- When an IP address cannot be converted to a name, *rusers*(1C) now prints the address in standard dot notation. (SCR 7615)

- The *uucp*(1C), *uucico(1M)*, and *cu*(1C) manual entries now document the read permission on */dev/ttyd** (required for *cu* and *uucp* to run.) (SCR 7829)

- The kernel IP layer now uses *arpwhohas*() to guard against duplicate IP address claims. (SCR 8130)

- In a previous release, *uucico* would sometimes dump core. This was caused by a divide by zero, and has been fixed. (SCR 8611)

- The *select*(2) manual entry has been changed to accurately reflect selectable devices.

- The *poll*(2) manual entry has been changed to include non-STREAMS devices. (SCR 8042)

- When shutting down a system, *last*(1) no longer lists users still logged in. (SCR 8639)

- The 4D1-3.3 version of *finger*(1C) now includes the same functionality as the BSD version. (SCR 8750)

- The *ndbm*(3B) manual entry now correctly states the allowed size of entries (1024 bytes). (SCR 8811)

- A bug in the DUART drivers that could cause the final block of data to be partially dropped has been fixed. (SCR 8933)
- Internet daemons *telnetd, rlogind,* and *ftpd* now print */etc/issue* before prompting for a password. (SCR 8936)

6.5 Bug Fixes to the Distributed Graphics Library

- With the improvements to the IRIX implementation of the *fork*(2) system call, the DGL will now work with substantially larger applications. (SCR 7771)

6.6 Bug Fixes to Mail Programs

- */bin/mail* (ATT mail) used to check for a lock file when called with the –e flag. Since the purpose of the lock file is to prevent corruption of the maildrop file by insuring that only one mail process can write to the mail file at a time, and since */bin/mail* called with the –e flag simply checks for the existence of mail in the maildrop file and returns without writing to it, any attempt to create a lock file in this circumstance is wrong. Not only is this behavior logically incorrect, it is potentially harmful since the existance of a legitimate (or even a bogus) lock file could cause f2/bin/mail –e to hang for several minutes.

  This behavior has been removed, and */bin/mail* –e no longer checks for lock files. (SCR 2708)
- */bin/mail* previously limited the name of the mail lock file to 13 characters. Since the name of the lock file is derived by appending *.lock* to the user's login name, this behavior created an effective (and unnecessary) limit of 8 characters for login names which could be used with the */bin/mail* program. The limit has been increased to 255 characters. (SCR 5727)

- */bin/mail* and */usr/sbin/Mail* previously ignored the existance of `Reply-To:` header fields in messages. RFC 822 requires that the contents of any such `Reply-To:` header field be used as the return address when replying to mail. Both mail programs have been modified to comply with the RFC 822 requirement. (SCR 5993)

- */usr/sbin/Mail* suffered from poor parsing of message list arguments to Mail commands (such as *print* and *delete*).

  Parsing of message list arguments has been corrected. Please see ''Specifying messages'' in the *mail_bsd*(1) manual entry for a complete description of supported message list argument formats. (SCR 7409, 7953)

- */usr/sbin/Mail* previously used relative pathnames when dealing with mail files other than the standard maildrop. Since the *Mail* program supports the *cd* command while running, the use of relative pathnames by *Mail* was an unwise practice which occasionally led to unfortunate results such as deleting the wrong mailfile.

  /usr/sbin/Mail has been modified to resolve all user-supplied pathnames to be absolute pathnames. This practice ensures that *Mail* will correctly identify the files it is dealing with. (SCR 7949)

- */usr/sbin/Mail* uses `From` lines in the mail file to determine the boundaries between messages. The standard format for such `From` lines is ''`From` *sender-address date*''. Messages containing `From` lines with null sender address fields but properly formed date fields are occasionally present on the Internet (and probably other places as well). The presence of such a message in a user's mail file would confuse the *Mail* program and cause it to consider the entire text of the offending message to be part of the preceding message in the mail file.

  /usr/sbin/Mail has been modified to consider `From` *date* to be a legitimate variation on the `From` line and will correctly detect the boundries of such messages. (SCR 8137)

- */usr/sbin/Mail* had numerous problems parsing RFC 822-style mail addresses. */usr/sbin/Mail* has been modified to correctly parse any RFC 822- compliant addresses. (SCR 8162, SCR 7456)

- */usr/sbin/Mail* did not treat comment lines (lines beginning with `#`) in the file *$HOME/.mailrc* correctly. *Mail* has been modified to ignore comment lines. (SCR 8196)
- A security hole in *sendmail* that allowed remote users to write to world-writable files has been fixed. (SCR 9157)
- */usr/sbin/Mail* now expands aliases in the `To:` field prior to determining whether to use the local signature (*.lsignature*) or the remote signature (*.rsignature*). (SCR 6843)
- */usr/sbin/Mail* now handles `From` lines with additional DST fields, such as MET. (SCR 7950)

6.7 Bug Fixes to Demos

- The *revolution* demo is no longer shipped. (SCR 5827, 5837, 6535)
- There was a visual problem in the *light* demo involving the border of the moving scene when rotating in MTV mode. (SCR 4135)
- Manual entries for the *dog*(6D) and *shadow*(6D) demos were previously not included. They are now included. (SCR 4383, 6491)
- Several manual entries for demos that weren't shipped with the system were being installed. These manual entries are no longer installed. (SCR 7763, 7952)
- There were several typographical and spelling errors in the information screens for the *radiosity* demo. These have been corrected. (SCR 8099)
- An error message previously appeared on the console when running the Buttonfly *Show Image* program. (SCR 8438)
- There was a bug in the old –T option of the *osview* program. *osview* no longer supports this option. (SCR 6187)
- An extraneous file (*/usr/demos/.workspace/database*) was being installed in the */usr/demos* directory. This has been corrected. (SCR 7451)

6.8 Bug Fixes to 4Dgifts

- The most recent version of *kermit* is now shipped as part of 4Dgifts. It allows you to transfer files over serial lines. (SCR 3581)
- Manual entries have been added for the following programs under */usr/sbin*: *gamcal, hist, icut, imged, imgexp, iset, istat, izoom, mousewarp, rle, scrsave, showci, verbatim}fl. (SCR 5830)*
- */usr/people/4Dgifts/examples/Fortran/Makefile* was zero length, causing a *make* executed at the top level fail. This has been corrected. (SCR 5851)
- The *4Dgifts* and *.4sight* files were owned by *bin*. Now they are correctly owned by *guest* (SCR 5990)
- *ipaste* did not work for image files which had a colormap type of CM_DITHERED. The images were displayed as though they were black and white. This has been corrected.

  On 4D series machines, *ipaste* was written to operate on RGB (24-bit) images or SCREEN (16-bit starting at zero) images but not on the old 3000 machine version of DITHERED (8-bit starting at some offset) images. See the NOTE regarding *fromdi* at the bottom of the *ipaste*(1G) manual page. (SCR 7433, 6060)
- The *goeml* type declaration in *4Dgifts/examples/grafix/curve3.c* has been corrected. (SCR 7701)
- *imged* is now installed with the *eoe2.sw.gltools* subsystem. (SCR 7712)
- The *dslib* manual entry incorrectly stated that the files *dstab.c* and dslib.c are located in */usr/people/4Dgifts/examples/devices/inquire.c*. The manual entry has been corrected to reflect their correct location in */usr/people/4Dgifts/examples/devices.* (SCR 8068)

- Errors in the way *libimage* handled calls to the *iopen* routine have been corrected. An error handling function was added to the *libimage* library whose default behavior is the same as the current, but which can be changed for programs that know how to deal with the various errors. There is now a routine called *i_seterror* that allows you to install your own *libimage* error handler. (SCR 8117)
- The *overlay* program in *4Dgifts* could leave graphics drawn in the overlay bitplanes if its window was moved or killed. This has been fixed. (SCR 9038)

6.9 Bug Fixes to Graphics

- The graphics subsystem was killed when too many *v3f* calls existed between bgnpoint and endpoint. (SCR 5431)
- *getcpos* has always returned the character position in absolute screen coordinates. However, its arguments used to be typed as `Screencoord *` implying a window-relative value. They are now `short *`.
- *greset* would try to load the first 8 color map entries of the color map for the current drawing mode. It now calls `drawmode(NORMALDRAW)` first.
- The *setmonitor*(3G) manual entry incorrectly listed the **HZ30_SG** option as **HZ30_SG**. This has been corrected. (SCR 3122)
- Incorrect FORTRAN references in the manual entries of the advanced lighting routines have been corrected. (SCR 3403)
- In a previous release, the *drawmode* manual entries incorrectly specified the argument to be logical. These manual entries now correctly specify the argument to be integer. (SCR 4282)
- On GT/GTX systems, *blink*(3G) with rate of `-1` did not turn off blinking as was documented in the manual entry. Specifying a *blink* rate of `-1` now turns blinking off. (SCR 4350)
- The *defpup*(3G) manual entry now lists the **%n** format option. (SCR 4353)

- Problems using `unqdevice(INPUTCHANGE)` have been corrected. (SCR 5548)

- The *defrasterfont* manual entry has been corrected to say that the ''offset'' field in the Fontchar structure is an index into the raster array, not a byte offset. (SCR 6040)

- Multiple calls to *blink* would crash the graphics pipeline. This has been fixed in the microcode. (SCR 6139)

- *getmcolor* used to work only in color index mode. It was fixed to work in RGB mode as well. (SCR 6621)

- Context switching sometimes caused screen mask clipping of characters to fail. This has been fixed. (SCR 7346)

- Negative arguments to *rectzoom* used to cause the graphics to crash. This has been fixed. (SCR 7348)

- When *picking* returns a large amount of data, it could overflow and trash the name stack. This has been fixed. (SCR 7783)

- Previously, backface polygon removal used an algorithm that did not work correctly in all cases. This has been fixed. (SCR 7831, 9237)

- *rectread* in double buffer mode on an 8-bit Personal IRIS used to return the wrong values. This has been fixed. (SCR 8468)

- In some cases, light source 0 would get unbound after drawing a scene. This has been fixed. (SCR 8697)

- On a Personal IRIS with an RE1 raster engine (`GR1.1` in the hardware inventory), a large *rectwrite* with a *rectzoom* factor greater than one used to fail. This has been fixed. (SCR 8699)

- On the Personal IRIS, *endfeedback* used to return `-1` if the feedback buffer overflowed. It now behaves as the other IRIS-4D models and returns the size of the buffer. (SCR 8709)

- Depthcued, flat shaded polygons previously displayed in a single color. Now the color is recomputed for each polygon. (SCR 8838)

- On a Personal IRIS with an RE1 raster engine (`GR1.1` in the hardware inventory), outlined polygons failed to reset the line stipple for the first line segment. This has been fixed. (SCR 8890)

- Backfacing polygons used to generate hits during picking. This has been fixed. (SCR 8910)
- After an *endpoint* command, vertex commands used to continue to draw points. This has been fixed. (SCR 8954)
- On a Personal IRIS with an RE1 raster engine (`GR1.1` in the hardware inventory), *rectwrites* in combination with screen mask clipping used to fail. This has been fixed. (SCR 8890)
- On a Personal IRIS with an RE1 raster engine (`GR1.1` in the hardware inventory), stippled lines with a width greater than one did not draw correctly. This has been fixed. (SCR 9073)
- *rectread* followed by *rectwrite* used to fail if the Z buffer was the *readsource.* This has been fixed. (SCR 9178)
- On a Personal IRIS with an RE2 raster engine (`GR1.2` in the hardware inventory), outlined polygons drawn with stippled lines with a width greater than one did not draw correctly. This has been fixed. (SCR 9228)
- On a Personal IRIS with Turbo graphics, clipping of large polygons (more than 40 vertices), would sometimes cause the graphics to crash. This has been fixed. (SCR 9253)
- On a Personal IRIS with Turbo graphics, drawing a polygon with no vertices would cause the graphics to crash. This has been fixed.

6.10 Bug Fixes to 4Sight

- The menus in 4Sight were too sensitive. Small mouse motions would make the pull right menus disappear. (SCR 4035)
- The ownerships of the files in */usr/lib/fmfonts* were changed from root/bin or root/sys to bin/bin. (SCR 4106)
- The *psview*(1) manual entry was corrected to match the functionality of the *psview* program. (SCR 4275, SCR 6048, SCR 6300, SCR 7289)
- A range check error test was added for the third parameter of the *getinterval* operator, to prevent server failure when that number is negative. (SCR 4428)

- The *say* program's unix process did not terminate when its window was killed. This was eliminated when some reference counting errors were fixed. Not reproduceable in 3.1D. (SCR 4434)
- The default */etc/hosts* file would attempt to route messages for the local *hostname* through the ethernet driver even when the driver wasn't *lboot*ed in the kernel. This would result in the 4Sight toolchests not appearing. (SCR 4510)
- A PostScript program with more *gsave*s than *grestore*s could crash 4Sight. (SCR 4819)
- Some graphics state operators, such as *setrgbcolor* would generate a typecheck error when given real numbers with more than 3 decimal places. (SCR 4873)
- Using very large (> 300 pixels high) characters could crash the NeWS server. (SCR 4942)
- Function invocation using `defpup("title %t %F|...)` did not work properly. (SCR 4982)
- If the *token* operator was given an executable array, the NeWS server crashed with a bus error. (SCR 5108)
- Drawing very small arcs using the *arc* or *arcn* operators crashed 4Sight. (SCR 5227, SCR 5228)
- Certain arcs drawn with the *arcn* operator were drawn incorrectly and crashed 4Sight. (SCR 5229)
- Drawing a dashed line from somewhere on the sceeen to a very large positive *x* coordinate crashed 4Sight. (SCR 5248)
- A program using *fprintf* with a large number of arguments crashed the 4Sight when it was run as a part of a long series of PostScript tests. (SCR 5300)
- Dragging whole windows rather than outlines caused problems on certain systems and could crash 4Sight. The crash has been fixed and 'Flip Drag' has been removed from the WindowChest because it doesn't work very well on certain 4D models. (SCR 5638)
- The thick rubberbanding style lines would break into 2 separate lines when the window was being resized. The thick lines weren't used at all when moving a window. (SCR 5744)

- The maximum number of windows that can be opened simultaneously was changed from 50 to 255 for the Personal Iris. (SCR 5920)
- 4Sight hung after running *psview*. (SCR 5962)
- Modifying a file while it was being displayed by *psview* hung the NeWS server. *psview* has been changed to terminate when the file it is displaying is changed. (SCR 6001)
- The *consize* function was unimplemented in 3.0 and 3.1. It is implemented and working in the environment manager release. (SCR 6006)
- While resizing a window, windows beneath the cursor could get input focus. This has been corrected. (SCR 6019)
- Running the NeWS horse race program from *comp.window.news* crashed the 4Sight when it was run 15-20 times. (SCR 6081)
- When a window's upper left corner was off the screen and the window was stowed, it's icon could be completely off the screen. (SCR 6188)
- System performance slowed considerably after using *psview* intensively. (SCR 6413)
- Attempting to open more than the maximum number of windows now results in a detectable failure. (SCR 6782)
- Running a program that *malloc*s all the available memory can crash the 4Sight. (SCR 6967)
- When in *overlay* text mode, *wsh* did not clear the *overlay* planes when stowed to an icon. (SCR 7213)
- 4Sight sometimes hung or terminated abnormally when *psview* was used to display PostScript files imported from external sources. (SCR 7241)
- Lines were not properly clipped in an imagebackground program on a Personal Iris. (SCR 7242)
- The *psview* process did not exit when the window process terminated. (SCR 7311)
- The default colormap is now documented in the *4Sight Programmer's Guide*. (SCR 7337)

6.11 Bug Fixes to the WorkSpace

- When a file is dragged to a new window where it overwrites an existing file, the new icon now lands where you put it (instead of at the bottom of the directory view). (SCR 7564)
- When *dirview* is used to open a directory that does not exist, the user is informed via a notifier box, rather than a message to the console window. (SCR 7627)
- WorkSpace now updates when a dangling symbolic link changes to point to a real file. (SCR 7695)
- WorkSpace list views now update when the date on a file changes (if they are sorted by date). (SCR 7758)
- There is a shell script in */usr/people/4Dgifts/examples/WorkSpace* called *shellDevice*, which can be placed in */etc/transferDevice*. You can then run the transferManager, select it as an active device, and access from the WorkSpace popup transfer rollover the ability to open a shell window on any selected directory. (SCR 8217, SCR 8394)
- If directory views are stowed when the WorkSpace is shut down, they will be stowed when it starts up again. (SCR 8254)
- Double clicking on the folder of a stowed directory view now unstows the directory view. (SCR 8264)
- Files mounted from NFS now show up even if they are not exported with *root* privileges, or with completely open *read* priveleges. (SCR 8759)
- When viewing as list and sorting by date, WorkSpace now resorts its list if the date on a file is changed (even if the file is only touched).
- If you request to remove a file and WorkSpace can't put it in the designated dumpster directory, WorkSpace asks you if you want the file removed completely. Previously, this also turned off your preference for using the dumpster, so that anything you removed from then on was permanently removed instead of being sent to the dumpster.
- On an *altdrag* twice to the same directory, the icon formerly disappeared although the file actually existed.
- Sort by name on WorkSpace now sorts by local name, not full path name.

- WorkSpaces *umask* is now set correctly, so that files created by WorkSpace mirror the users *umask.* They used to be set to completely open permissions.
- You can now select more than 50 files and drop them on an executable.
- If the typer cannot type an item correctly, it now identifies it as unknown, instead of as a random type.
- There were several performance improvements in moving files around between directories.
- If you own a directory and its contents, you can drag it or remove it even if you do not have write permission on its contents. (This supports the same behavior as the shell.)

6.12 Bug Fixes to the X Window System

- The Xsgi man page has been redone. Further, we use traditional X man pages from the MIT tree. All X man pages and installation have been reworked. (SCR 5040, 7848)
- Xsgi now reads fonts in a traditional X manner. It can read X fonts as well as fontmanager fonts. Several fontmanager interaction and *lsfont* interaction bugs have been fixed. (SCR 5549, 8598, 8599)
- Xsgi now performs pixel-perfect rendering in all known cases. This involves a necessary performance loss in certain arenas. (SCR 7307, 7312, 7867, 8706)
- X performs an `ld` –r on all internally built libraries, shrinking their size considerably. There is no functional change. (SCR 7851)
- X header files have not changed and are consistent with the standard MIT distribution. (SCR 8224)
- Xsgi startup and server defaults are configurable from the command line or from NeWS startup via xstart(1). Note that xstart(1) is intended for use from NeWS, not from the command line. (SCR 8370)
- X sends correct keypress events for Alt keys. (SCR 8433)

- Xsgi uses SystemV *utmp*(4), which fixes the bug. (SCR 8868)
- Xsgi default visual type is PseudoColor. Xsgi exports two visuals: PseudoColor and StaticColor. Which is the default is configurable from the command line. (See SCR 8370, above.) Programmers who create colormaps matched to (nondefault) visuals should ensure that their windows and colormaps have the same visual type. (SCR 8944)
- XPutImage has been repaired in the server. Also, primitives which descend to SetSpans to a window now correctly render in all known cases. SetSpans is completely hidden from the programmer. (SCR 9275)

7. Known Problems and Workarounds

This chapter describes known problems with the 4D1-3.3 release and ways to work around them.

7.1 IRIX

- When the system is put in single user mode, the textport window on the workstation console may be put in a mode where keys typed on the keyboard are not echoed on the screen. If this happens, type:

  ```
  <Ctrl-j>stty sane<Ctrl-j>
  ```

- Some systems may not have enough room in their disk volume header to install the kernel debugging option, *dev.sw.debug*. If installation of the *dev* option fails with a message about space problems in the volume header, the *dev.sw.debug* subsystem should be removed.

- The *ioctl* command TIOCNOTTY does not work the same way on IRIX as it does on 4.3BSD. On both types of systems, this *ioctl* disassociates the calling process from its controlling terminal, if it has one. This *ioctl* is only effective when called on the file descriptor of a terminal device, but the file descriptor need not be associated with the controlling terminal of the process.

  In IRIX *ioctl*(TIOCNOTTY) also makes the calling process a *process group leader*, meaning that its process group ID after the *ioctl* is equal to its process ID. On a BSD system, *ioctl*(TIOCNOTTY) sets the process group ID of the calling process equal to zero, meaning that it is not a member of any process group. In a future release, the TIOCNOTTY command in IRIX will be changed to behave as the BSD version does. In the interim, the following workaround will give the BSD behavior:

```
ioctl(fd, TIOCNOTTY, NULL);       /* lose controlling tty */
BSDsetpgrp(0, 0);                 /* make sure pgrp == 0 */
```

- Password aging does not work with *pandora.* If you want to use password aging, visual login (*pandora*) must be turned off. Also, *pandora* ignores secondary (dialup) passwords.

- The printed manual pages for release 4D1-3.3 include a manual entry for *mktime*(3), but there is no corresponding on-line manual entry. The routine *mktime* is not provided in this release.

7.2 Program Development Tools

- The one-line C program

  ```
  s = ((p()) == 5) ? 6 : 4;
  ```

  coredumps *ccom* without producing an error message. The `?:` operator, combined with undefined variables (an erroneous external data definition) is the problem. Correct the program and the problem will go away.

- A function call through an integer constant coredumps *ccom.* An example of such a call is

  ```
  (((*(())0x40000)();
  ```

  The workaround for this problem is to put the address in a function pointer variable and indirect through the variable.

- Putting code in #include files can produce applications which are hard to debug with *dbx*(1), since the symbol table does not support nested includes of code. Putting declarations and data in #include files is, of course, acceptable.

- The *cc* options **–t**, **–h**, and **–B** do not work properly. Normally one does not need to use these options.

- Accidental substitution of a label name for a variable causes the C compiler to emit a cryptic internal error rather than a useful diagnostic. For example

  ```
  forced:    if(forced)  ....
  ```

  produces the error message:

  ```
  libmld: Internal: st_pdn_idn: idn (12) less than 0 or greater than max (5).
  ```

  The program is erroneous.

- In rare circumstances, a complex expression involving doubles can, if **–float** is supplied to *cc*(1), gives the message

  ```
  nop must be inside .set noreorder section
  ```

  Since **–float** is never required, do not use **–float** if this happens.

7.3 Networking

- The network startup script sets the IRIS's host ID to its Internet address with the *hostid*(1) command. Do **not** change the host ID to some other value — RPC-based programs assume the ID is an Internet address and they will likely fail if it is not.

- The CMC ENP-10 Ethernet controller is disabled for approximately 1–2 seconds when its multicast address filter is updated. This affects IP multicasting code that tries to send packets immediately after joining or leaving a multicast group with the IP_ADD_MEMBERSHIP and IP_DROP_MEMBERSHIP *setsockopt*(2) calls. *send*(2) or *write*(2) calls will return EBUSY while the controller is updating the filter. A suggested work-around is to check for EBUSY on output or add a delay after the *setsockopt*(2) call by calling *sleep*(3).

7.4 Graphics

7.4.1 Graphics Library

- `lmcolor`

  If a program goes into an `lmcolor` mode that affects material and then issues drawing commands, without ever having issued a color command, the drawing is done with an undefined material.

  Thus, after going into an `lmcolor` mode that affects material, programs should always issue some color-setting command before doing any drawing.

- *hollow polygons*

  On the PowerVision system, hollow polygons are drawn using the stencil bitplanes. Note 3 on the bug list of `polymode`(3G) manual page should read

  The stencil function must be set using

  stencil(1, 1, SF_EQUAL, 1, ST_KEEP, ST_KEEP, ST_KEEP);

  On future releases, polymode(PYM_HOLLOW) will set up the stencil function automatically.

- *rectcopy performance*

  Because of a bug, the current `rectcopy` performance on PowerVision systems never exceeds 4.5 Mpix/sec. This will be fixed in the next release.

- *quadword-aligned data*

  On Power Series models, a number of Graphics Library calls should be called with quadword-aligned data. The version of *malloc* in the *mpc* library (available by compiling with **–lmpc**) always returns quadword-aligned data. If these routines are called with non-quadword-aligned data, performance can be severely degraded. The routines affected are:

```
c3f, c3i, c3s
c4f, c4i, c4s
```

```
n3f
v2d, v2f, v2i, v2s
v3d, v3f, v3i, v3s
v4d, v4f, v4i, v4s
```

Also, on IRIS-4D/120 systems **only**, when calling these routines with non-quadword-aligned data or double-precision floating-point data, you cannot modify the data after issuing the call, until after a call to one of the following:

```
endclosedline
endline
endpoint
endpolygon
endqstrip
endtmesh
```

On Power Series models, data buffers for pixel rectangle transfer commands should be locked in memory using the *mpin* system call. If data is not locked, performance will be degraded. The routines affected are:

```
rectread
rectwrite
lrectread
lrectwrite
```

When using the *rectread* routine, the data buffer must be 32-bit aligned and the rectangle width must be given.

- *signal handlers*

  Do not make graphics calls within signal handlers. A single Graphics Library call may issue a sequence of commands to the graphics hardware. If a single handler were to issue graphics calls, it could disturb such a seqeunce and cause unpredictable behavior.

- *z-buffer inaccuracy*

  On VGX systems, the same primitive rendered in different modes may have different Z-buffer values. For example, a textured polygon and the same polygon rendered without texturing may have different Z-buffer values.

- *cyclemap*

  The Fortran wrapper for the `cyclemap` Graphics Library routine now expects arguments of type `integer*4`, as the *cyclem*(3G) manual entry documents.

- *point clipping*

  Points drawn on VGX systems while lighting, texture mapping, depthcueing, and user-defined clipping planes are disabled may clip incorrectly. Clipping is corrected if any of these modes is enabled.

- `lsetdepth`, `lshaderange`, and `lRGBrange`

  Programs that call `lsetdepth`, `lshaderange`, or `lRGBrange` should always explicitly set the GLC_ZRANGEMAP mode by calling `glcompat`, because the GLC_ZRANGEMAP mode influences their behavior, and its default value is different on different systems.

7.4.2 GT/GTX Lighting Hardware

In rare cases, the GT/GTX lighting hardware exhibits a numerical precision problem that causes the specular component to flicker on and off. This only happens when the normal, the light direction, and the viewer are oriented in a certain way.

Specifically, when the light direction is exactly perpendicular to the normal, there is a chance that the specular component may flicker. Users trying to bind a light to an object are much more likely to encounter this symptom.

The reason for the flicker is that lighting makes a decision based on which side of a plane (as defined by the normal) a light is. When a light is placed exactly in this plane, slight variations in numerical precision cause this decision to change.

A workaround for this problem is to move the light ever so slightly out of this plane. A very slight rotation of the light is sufficient. For example, if the light position is (0.0, 0.0, 1.0) and the normal is (1.0, 0.0, 0.0) it could be changed to (-0.0017, 0.0, 0.9999). This is a 0.1 degree change which should not affect the appearance at all.

7.5 Windowing

7.5.1 NeWS Menus in Application Windows

In a multiple-window program, the entry in the ''Quit'' part of the pop-up menu represents the application name. By convention, this is the *name* argument to the first *winopen* issued by the program. This is the correct behavior.

7.5.2 WorkSpace

Server Problems

Starting WorkSpace Automatically

WorkSpace may sometimes fail to start upon login if ''autoworkspace'' is enabled and you use the Yellow Pages (YP) services. This problem occurs if the YP server is not yet bound when the system tries to start WorkSpace. If this happens, wait a minute or two, and run WorkSpace from the system chest or from the command line. Use the *ypwhich*(1) command to know when a server has been bound and to show which server supplies Yellow Pages services to your system.

If you are using a YP master server that is not an IRIS-4D running Release 4D1-3.2 or 4D1-3.3, you need to make add some entries to the ''rpc'' map in order for WorkSpace and the System Manager to run successfully. On the YP master server machine, edit the */etc/rpc* file, and add the following two lines:

```
sgi_toolkitbus   391001
sgi_fam          391002
```

Then, also on the YP master server, execute the following commands:

```
cd /usr/etc/yp
make rpc
```

(These commands to make and push the YP map may differ on non-IRIS systems.) After the changes have propagated to your system's YP server, check for the entries using the command:

```
ypcat  rpc  |  egrep  'sgi_fam|sgi_toolkit'
```

When both entries appear, you can start WorkSpace and the System Manager.

If these steps are not taken, and you are running with a foreign server, WorkSpace will not appear, but will instead print the following error message:

```
Can't connect with file access monitor.
```

View Updating Problems

- view updating on a local system

  On rare occasions, WorkSpace will not reflect changes in the file system (for example, a running application won't stand up, or changes won't be reflected in a directory.) The same bug may cause *vadmin* to report that a tool is running when in fact it is not. To correct this situation, quit and restart *vadmin.* In addition, *mailbox* may appear to stop receiving mail (that is, stop raising the flag on the mailbox). To correct this situation, quit and restart *mailbox.* If this happens, make a link to the offending file or directory and then remove the link. Closing and re-opening the offending directory will also make WorkSpace start noticing file system changes again.

- view updating over NFS

  WorkSpace maintains an up-to-date view of files on the local system. However, when it views files over NFS, it polls to track the file changes made on the remote systems. This polling interval is normally set to three seconds.

Should you wish to change this to some number greater than 0 and less than 10, edit */usr/etc/inetd.conf.* Change the line referring to `fam` by adding **–t** *n* to the end of it, where *n* is the number of seconds specified for the polling interval. For example, to set the polling interval to 5 seconds, change the line in *inetd.conf* as follows:

```
sgi_fam/1  stream  rpc/tcp wait  root  /usr/etc/fam  famd -t 5
```

Then, as *root*, tell *inetd* to re-read *inetd.conf* with the command

```
killall -HUP inetd
```

Wait a few seconds, and restart WorkSpace.

7.6 4Sight

Under some conditions, a window border may remain on the screen after its program has terminated. To minimize this visual inconvenience, either push the window to the bottom of the window stack by selecting the "Push" window menu button, or iconify the window by selecting the "Stow" window menu button (or by picking the stow icon in the window border). The window border also goes away if the user logs out of the Window Manager with the "Log Out" button and logs back in again.

8. Documentation Notes and Errors

This chapter contains corrections to documentation errors and additional notes for this release.

Note: Generally, on-line documentation, such as the *man* pages is more up-to-date than the printed hard copies. This is not true for this document.

8.1 Documentation Errors

8.1.1 IRIS-4D Series Owner's Guide

Version 4.0 of this manual contains these errors:

- Page 5-137 contains the reference ''See Appendix A for instructions on *vi*.'' This reference is incorrect. The *vi* text editor is explained in the *vi* on-line manual entry.
- Page 3-56 contains a typographical error in the section ''How to change Existing Permissions''. The ''='' sign in the example:

  chmod *who=permission file(s)*<**return**>

  should be a dash, as follows:

  chmod *who-permission file(s)*<**return**>

8.2 Making an ASCII Terminal the Console

Section 9.2, ''Making an ASCII Terminal the Console'' of the guide *Writing Device Drivers for Silicon Graphics Computers* describes how to set up your system with an ASCII terminal functioning as the console. The documentation should include a warning that the terminal must be tested before you run: **`setenv console d`**

If you do not test the terminal first to make sure it works, and the terminal does *not* work, you will be unable to respecify the console and use your system. To avoid this situation, test the terminal before setting the console environment variable.

An easy way to do this is to try the terminal out on port 2. Edit the file */etc/inittab* so that the line that begins with `t2:` looks like this:

```
t2:23:respawn:/etc/getty ttyd2 co_9600          # port 2
```

If you needed to change this line in */etc/inittab*, after saving the changes

1. type **`telinit q`** at the shell prompt

2. plug the terminal into port 2

   If this line in */etc/inittab* already looked like this, plug the terminal into port 2.

   If the terminal works, continue on with the instructions in Section 9.2. If it does not, replace the terminal with one that works.

8.3 Installing an Optional Floppy Drive

Some important steps in the procedure for installing an optional floppy drive are missing in the optional floppy drive documentation. These include several things that need to happen before you can access the floppy drive.

1. After physically installing the drive and rebooting, run *hinv* and make sure the system knows the floppy is installed.

2. Edit the */usr/sysgen/system* file and uncomment the entry

   ```
   *INCLUDE: smfd
   ```

   by removing the leading asterisk.

3. Create a new kernel by either using */etc/init.d/autoconfig* or by using *lboot* and moving the new *unix.new* into */unix*, making sure you move the current */unix* to *unix.old*. The problem with *autoconfig* is that it doesn't make a copy of the current kernel.

4. Reboot the system.

5. Next, verify that the device files that will support this floppy exist. These files are:

   /dev/rdsk/fds0d[*1-7*].[*48 96 96hi*]

   If they do not exist you can create them by typing the appropriate command from the following list for the type of drive you are installing:

   /dev/MAKEDEV fds0d*[1-7]*.96hi = doublesided/hidensity
   /dev/MAKEDEV fds0d*[1-7]*.96 = doublesided/doubledensity
   /dev/MAKEDEV fds0d*[1-7]*.48 = doublesided/lowdensity

If, for some reason, MAKEDEV does not create them, you can make them manually using the *mknod* command. Since the floppy drive is typically addressed as `unit 3`, the commands for that particular configuration are listed below. Become super user and enter:

```
mknod /dev/dsk/fds0d3.48 c 40 96
mknod /dev/dsk/fds0d3.96 c 40 97
mknod /dev/dsk/fds0d3.96hi c 40 98
```

It is possible that the floppy will be addressed as something other than `unit 3`. If so, the above command is the same except for a change to the last argument or minor number. The formula for determining what that number should be is as follows:

```
32 * <unit #> + 0 = fds0d<unit #>.48
32 * <unit #> + 1 = fds0d<unit #>.96
32 * <unit #> + 2 = fds0d<unit #>.96hi
```

Note: The floppy driver now supports 3.5-inch floppy drives.

To use a floppy under SoftPC/AT*, you must have a device called */dev/pc_floppy* that needs to be linked to one of the above devices, depending on the type of floppies you will be using (not the device that it is currently linked to).

This floppy is also completely functional under */unix* and can be accessed with *tar*, *cpio*, *bru*, *dd*. You can create file systems on floppies and mount them if you wish.

To make use of the floppies in */unix*, format them with *fx* –x and follow this procedure:

1. Select the device type `fd, controller # 0`, and the appropriate drive number (typically `3`).

   fx then prompts for a drive type (`48 96 96hi 480`). (Select the appropriate type according to the above instructions.)

2. Select *format* from the *fx* menu.

   It will take a minute or two to format.

3. Exit *fx* and then begin using the floppy.

Note: Floppies formatted under *fx* are not compatible with SoftPC/AT and vice versa. In addition, SoftPC 1.1 does not support local or real floppy drives. You must have SoftPC/AT 2.0.

SoftPC/AT is a separate product, not an update to 1.1. Contact your sales representative for more information about SoftPC/AT.

8.4 Graphics Library Programming Guide

All of the code examples from the *Graphics Library Programming Guide* are now available on-line, in the directory */usr/people/4Dgifts/examples/glpg*.

Appendix A: X Window System Execute-only Environment

This appendix contains release notes for the X Window System™ execute-only environment for Silicon Graphics, Inc.'s IRIS-4D Series systems.

The main topics covered in this chapter are:

- Overview
- How to run the X Window System
- Fonts
- Miscellaneous Changes
- Known Problems

The X Window System server provides an interface for programs written for the X Window System, Version 11, developed jointly by MIT's Project Athena and Digital Equipment Corporation, with contributions from many other companies. It was designed by Robert Scheifler, Philip Karlton, Jim Gettys, and others at MIT, and was based in part on the ''W'' windowing package developed by Paul Asente at Stanford University.

The X Window System was designed to provide a distributed, standard window system for the UNIX workstation community. It is provided in 4Sight so that you can easily port programs written for the X Window System environment, or run remote X Window System clients on your IRIS screen. The X Window System uses a software-intensive 2-D imaging model, and thus is not appropriate for developing real-time 3-D applications for the IRIS-4D Series workstation.

These release notes describe the X11 implementation of the X Window System, available for use with release 4D1-3.3.

A.1 Overview

This release of the X Window System is organized into several components:

- Execution-only environment. This consists of the X server and some essential clients and configuration files.
- X fonts. These fonts are from MIT and will be found in /usr/lib/X11/fonts.
- X applications. Most clients, including those from contrib, fall into this category.
- X demos. Ico, puzzle, maze, muncher, etc.
- Development option. This option contains libraries and include files, as well as utilities such as imake, so that a user can create (i.e. compile and link) X clients.

All of the above components except for the development option are contained in the Exectution Only tape set.

The purpose of this document is to describe the execution-only component, fonts, and generally how to use X on an SGI platform. This document describes all the components except for the development option, which is described by another document.

A.1.1 X Window System Documentation

The execute-only environment includes the following documentation:

- these release notes
- manual pages for the X Window System server and clients

It is strongly recommended that you read the *X Window System User's Guide for Version 11*, by Tim O'Reilly, Valerie Quercia, and Linda Lamb (O'Reilly & Associates, ISBN 0-937175-29-3). Although these release notes give you all the information required to run X Window System clients, the *X Window System User's Guide* contains additional information. (See section A.6 for information on obtaining the *X Window System User's Guide*.)

Also, you can obtain the X11R4 release directly from MIT. The MIT release includes a substantial amount of documentaion about X, plus X source. To obtain materials directly from MIT, contact:

MIT Software Distribution Center
Technology Licensing Office
MIT E32-300
77 Massachusetts Avenue
Cambridge, MA 02139

The MIT ''X Ordering Hotline'' is (617) 258-8330.

A.2 Running the X Window System

The X Window System runs on the same screen as the 4Sight window system.

A.2.1 Starting Xsgi

In release 4D1-3.2, the X server (Xsgi) was started by inetd, and it was necessary to change inetd.conf in order to change the default Xsgi options.

With release 4D1-3.3, Xsgi is no longer started by inetd. Instead, the new program *xstart* gets executed during 4Sight startup. *xstart* starts up Xsgi provided that *xSGINeWS* is configured to be ''on.'' By default, *xSGINeWS* is configured ''on'' after an install of the X11 Execution Environment (eoe2.sw.X11).

For information about how to configure an item such as *xSGINeWS*, see the *chkconfig(1)* man page. For information about how to start Xsgi with different options besides the defaults, see the *xstart(1)* man page. The *xstart(1)* man page also describes how to configure xSGINeWS on a per-user basis.

A.2.2 Running Clients

Before an X client can be run, the server, Xsgi, must already be running. If the X server is not running, then an attempt to execute an X client will result in the error message "Error: Can't Open display" on stderr. (See above regarding how the X server is started.)

Note: the error message "Error: Can't Open display" can be produced by other errors besides the X server not running. For instance, an incorrect or non-existent DISPLAY environment variable (see below) will also cause this error.

DISPLAY Environment Variable

Most X clients allow you to supply an argument ''-display host:serverno'' in order to specify the machine on which the client will actually be displayed. However, in this absense of this argument, the DISPLAY environment variable will generally determine where the client's windows will appear.

One way to set DISPLAY is by setting it to the string ''machine:0'' where machine is either ''localhost'' or the current hostname. For example, to do this on a host called fireside, type the following command in an IRIX (csh) shell:

```
setenv DISPLAY fireside:0
```

A more general way of setting DISPLAY is to set it to the string '':0''. If an X client tries to connect to the display '':0'', a connection will be made on the local host.

Remote Host Access

If you wish to make your IRIS-4D Series workstation accessible to others so that they can run X Window System clients remotely from your IRIS, follow this procedure:

1. Make sure that at least one X Window System client is running on your IRIS.

2. At an IRIX shell, type the command:

```
xhost +
```

This sets up your X server's access privileges so that other remote machines can access it. See the man page for *xhost* for more information.

Note: When the server detects that there are no clients running, it resets the list of remote machines permitted access to it. Thus, from the time before xhost is run, up until the remote access is attempted, there should be at least one X client running; e.g., xlogo and xclock.

A.2.3 When X should be run without Backing Store

Avoid Backing Store if no X window manager

If an X window manager is not running, there are situations for which it will be impossible for X backing store to work correctly.

Thus, if you do not run an X window manager, it is recommended you run Xsgi with the **-bs** option. Note that this is exactly what *xstart* does by default. (See the *xstart(1)* man page.)

Avoid Backing Store on 4D/50G through 4D/80G systems

The performance and functionality of the X Window System on the 4D/50G through 4D/80G (non-GT and non-GTX) platforms is less than that of the other product lines due to hardware limitations. As a consequence of this, the performance of backing store is very slow.

On these systems, therefore, the X server should be brought up without backing store. As mentioned above, this is what *xstart* does by default.

A.2.4 Specifying the Default Visual for the X server

The following options to Xsgi specify the default visual of the X server:

-static	Static visual. All 256 colors in default colormap are pre-allocated.
-envm	Pseudo-color visual, with only the gl color indices 16-31 available to X as writable color cells. This provides pseudo-color with the least interference to 4Sight.
-gl	Pseudo-color visual, with the gl color indices 16-31 and 64-255 available to X as writable color cells. This gives 208 writable colors for X, and still prevents X from interfering with the default (gray) 4Sight border colors.
-pseudo	Pseudo-color visual, with all of the gl color indices 0-255 available to X as writable color cells.

The default arguments used by *xstart* for starting Xsgi are "**-bs -envm.**" The *xstart(1)* man page describes how these defaults can be overridden.

A.3 Fonts

In release 4D1-3.2, the Xsgi X server only supported SGI's fontmanager-style fonts, the same ones as used by SGI's 4Sight windowing system. Now in release 4D1-3.3, Xsgi supports these fonts, but it also supports pure X-style fonts. The default X font path is:

/usr/lib/X11/fonts/misc/,
/usr/lib/X11/fonts/100dpi/,
/usr/lib/X11/fonts/75dpi/,
/usr/lib/fmfonts/

That is, once Xsgi is running, it is able to read the fonts in any of the above directories.

Most of the fonts which SGI supplies in /usr/lib/X11/fonts are ''compressed snf,'' where ''snf'' stands for ''server natural format'' according to X terminology.

The fonts supplied in /usr/lib/fmfonts are 4Sight fontmanager-type fonts.

The program *xlsfonts* will produce a list of all the fonts the X server is able to read from the current fontpath.

Note: The only fonts supplied with the X execution-only component of this release are a minimal set of fontmanager-type fonts, and they're found in /usr/lib/fmfonts. To get the X fonts in /usr/lib/X11/fonts, it is necessary to install the eoe2.sw.Xfont component of this release.

A.3.1 Adding New Fonts

- After you add a new font (to one of the X fontpath directories), the X program *mkfontdir* must be run in the directory containing the new font. This must be done for any (even fontmanager) new font(s), before X will recognize the new font(s).
- If you add a fontmanager-type font to the system, note that the X server expects to find values for the properties Font Ascent and Font Descent. If you do not include these values in your font description file, the server cannot function properly.

A.3.2 Font Naming Conventions

The *xlsfonts* program can be used to list all of the fonts that are found in font databases in the current font path. Font names tend to be fairly long; however, the X server supports wildcarding of font names, so the full specification

```
-adobe-courier-bold-r-normal--14-100-100-100-m-90-iso8859-1
```

could be abbreviated as

```
-*-courier-bold-r-*--14-100-*-*-*-*-*
```

or, more tersely:

```
-*-courier-bold-r-*-14-*-100-*
```

The X client *xfontsel* may also be helpful in finding valid X font names.

Fontmanager fonts are specified by a name of the format *name.point_size*, where ''name'' is case insensitive.

The X client *xlsfonts* will list allowable names for both types of fonts.

A.3.3 Miscellaneous differences in fonts since 4D1-3.2

- Some fonts, such as New Century Schoolbook, were supported in release 4D1-3.2 by the fontmanager interface and resided in /usr/lib/fmfonts, but now in release 4D1-3.3 are pure X fonts and reside in /usr/lib/X11/fonts.
- In release 4D1-3.2, a font was specified by ''name.size'' where ''size'' referred to point size. In release 4D1-3.3, ''name.size'' is simply an allowable format for the name of the font.
- In release 4D1-3.2, if ''name'' existed, but ''name.size'' did not, the fontmanager provided a font closest in size to name.size. However, in release 4D1-3.3, if a client makes a request for the font ''name.size'' and this font doesn't exist, then Xsgi will return an error to the client saying that font doesn't exist.

  When a requested font does not exist, it is up to the client as to how to proceed. Many clients give an error message such as

  ```
  Warning: Cannot convert string "xxx" to type FontStruct
  ```

  and will then use the font ''fixed'' instead of the font originally requested.
- In release 4D1-3.2, requesting Helvetica.11 *worked* even though this font didn't exist. In release 4D1-3.3, it's necessary to ask for a font that exists, such as Helvetica.10 or Helvetica.12.
- The file */usr/lib/fmfonts/fmaliases* is no longer used. The current mechanism for font aliasing is by a *fonts.alias* file in a given font directory. The format of this file is fully described in the *mkfontdir* man page.

A.4 Miscellaneous Changes

- Now, the drawing of lines and other X graphical objects is done to exactly those pixels on the screen as specified by the X protocol.

A.5 Known Problems

- If an X window manager is not running, then there are situations for which it will be impossible for X backing store to work correctly.
- The X color data base /usr/lib/X11/rgb.txt is tuned to gamma 1.0. That is, if an X program uses the colors from rgb.txt, those colors will appear ''washed out'' unless the user has executed the command

  ```
  gamma 1.0
  ```

  However, if Xsgi is given the -pseudo option, then it will set the gamma appropriately, and it's not necessary for the user to manually execute the gamma command.
- The **<Caps Lock>** key does not work at all. The implementation of 4Sight is such that it's impossible for any Graphics Library application, including X, to query the current up/down state of a key. Since X can't reliably know the state of Caps Lock, it completely ignores this key.
- X is not always able to perfectly track the state of some modifier keys, such as the control key.
- xmh will not work unless mh is also installed on the system. However, mh is not supplied by SGI.
- ''xterm -C'' (to run xterm as a console) doesn't work. The ''-C'' is ignored.
- Xsgi does not support the controlling of the following:

  mouse acceleration
  bell volume
  keyclick

  either by command-line options to Xsgi, or through xset. (However, mouse acceleration can be controlled by the GL tool *mousewarp*, executed from the shell.)

- The program *xditview* (in this release) will not work with the */usr/bin/troff* which was supplied with SGI's 4D1-3.2 dwb release.
- If, *before* starting up twm, you first start up an X client and then iconify it (under 4Sight), this client will be lost after you later start twm. That is, after twm is started, the client won't be visible (in either its iconified or uniconified state). Also, the client doesn't reappear after twm is exited.

A.5.1 4Sight Anomalies

- If an X window manager is not running, it's impossible for the X server to know that only part of an X window has been damaged. This is because X must depend on the GL REDRAW event, which applies only to a whole window.

  Thus, if the user is not running an X window manager, and if just part of an X window is damaged, then Xsgi will generate expose events for the entire window plus all of its subwindows. The result is extensive window redrawing after minor window damage.
- If backing store is used for an X window, and if an X window manger is not running, then after window damage, Xsgi will repair the damage and also send an expose event to the client.

  Note: An X server is never required to provide backing store. At any time, the server is allowed not to provide it. Once backing store is provided, the server may stop providing it at any time. Thus, clients should always be able to repair window damage in response to expose events.
- When an X window manager is started up, there's lots of flashing. When an X window manger is exited, it may take several seconds for the X root window to fully disappear.
- If an X window is killed by clicking on the ''lightning bolt'' from the 4Sight window frame, the X server kills the the entire X client, not just the X window.

A.6 For More Information

For more information about using the X Window System, see the *X Window System User's Guide for Version 11* by Tim O'Reilly, Valerie Quercia, and Linda Lamb (O'Reilly & Associates, ISBN 0-937175-29-3). This publication is available from Silicon Graphics, Inc. Contact your sales office for more information.

Appendix B: Sphere Library Reference Manual Pages

This appendix contains the C and FORTRAN manual pages for the Sphere Library.

The C manual pages precede the FORTRAN manual pages.

C	FORTRAN
LIBSPHERE (3L)	LIBSPHERE(3L)
SPHDRAW(3L)	SPHDRA(3L)
SPHFREE(3L)	SPHFRE(3L)
SPHGNPOLYS(3L)	SPHGNP(3L)
SPHMODE(3L)	SPHMOD(3L)
SPHOBJ(3L)	SPHOBJ(3L)
SPHROTMATRIX(3L)	SPHRMA(3L)

NAME

libsphere – the Sphere Library

SYNOPSIS

#include <gl/sphere.h>

```
int sphdraw(params)
float params[4];

int sphgnpolys()

void sphfree()

int sphmode(attribute, value)
int attribute, value;

void sphobj(objid)
Object objid;

void sphrotmatrix(mat)
Matrix mat;
```

DESCRIPTION

The Sphere Library renders spheres by issuing GL calls. **sphdraw** draws a sphere. **sphmode** sets various attributes that affect the speed and quality of spheres rendered by **sphdraw. sphrotmatrix** allows you to control the orientation of spheres by providing a rotation matrix. **sphgnpolys** returns the number of polygons per sphere, in the mode currently selected by **sphmode. sphobj** operates like **sphdraw**, except that, instead of immediately rendering a sphere, it creates and returns a GL object, which can be rendered using **callobj**.

USAGE

To link a program with libsphere, specify **-lsphere** on the compile line. When linking a program with libsphere, you must also specify the Graphics Library, **-lgl_s**, and the Math Library, **-lm**.

CAVEAT

In the current implementation, the Sphere Library maintains an internal cache of GL objects, one for each combination of sphere attributes for which **sphdraw** or **sphobj** has been called. The first call to **sphdraw**, or **sphobj**, with a combination of attributes that has not yet been rendered will result in the creation of a new GL object and, thus, will take longer than subsequent calls with the same combination of sphere attributes.

If your program uses the Sphere Library and also manipulates its own GL objects, you must be careful that your allocation of object identifiers does not conflict with the Sphere Library's allocation of object identifiers for its internal cache. The Sphere Library calls **genobj** for its object identifier allocation. If your program allocates object identifiers for its own use, you should either use **genobj** to allocate object identifiers or use **isobj** to verify that any object identifiers your program generates are not already being used by the Sphere Library. See *makeobj*(3g) and *genobj*(3g) for further discussion of object identifiers.

If you need to free the virtual memory used by the Sphere Library's internal cache, you can call **sphfree** to free all its GL objects and their associated memory.

SEE ALSO

sphdraw, sphgnpolys, sphfree, sphmode, sphobj, sphrotmatrix

NAME

sphdra – draw a sphere

FORTRAN 77 SPECIFICATION

integer*4 function sphdra(params)
real params(4)

PARAMETERS

params

a 4-element array of reals containing the size and location of a sphere. The first 3 elements are the the x, y and z coordinates of the sphere center, followed by the sphere radius.

FUNCTION RETURN VALUE

This function returns -1 if an error occurs, otherwise 0.

DESCRIPTION

sphdra draws a single sphere of specified radius with center at the specified location according to the sphere attributes set by **sphmod.** The sphere is rendered as polygons with a normal at each vertex, and thus will be lighted if lighting is enabled.

SEE ALSO

libsphere, sphfre, sphgnp, sphmod, sphobj, sphrma

NAME

sphdraw – draw a sphere

C SPECIFICATION

int sphdraw(params)
float params[4];

PARAMETERS

params

a 4-element array of floats containing the size and location of a sphere. The first 3 elements are the the x, y and z coordinates of the sphere center, followed by the sphere radius.

FUNCTION RETURN VALUE

This function returns -1 if an error occurs, otherwise 0.

DESCRIPTION

sphdraw draws a single sphere of specified radius with center at the specified location according to the sphere attributes set by **sphmode.** The sphere is rendered as polygons with a normal at each vertex, and thus will be lighted if lighting is enabled.

SEE ALSO

libsphere, sphfree, sphgnpolys, sphmode, sphobj, sphrotmatrix

NAME

sphfre – free all internal sphere objects in the Sphere Library's internal cache

FORTRAN 77 SPECIFICATION

subroutine sphfre()

PARAMETERS

none

DESCRIPTION

sphfre will free all of the sphere objects in the internal cache of spheres maintained by the Sphere Library. Future calls to **sphdra** and **sphobj** will re-create and re-cache sphere descriptions as needed.

SEE ALSO

libsphere, sphgnp, sphdra, sphmod, sphobj, sphrma

NAME

sphfree – free all internal sphere objects in the Sphere Library's internal cache

C SPECIFICATION

void sphfree()

PARAMETERS

none

DESCRIPTION

sphfree will free all of the sphere objects in the internal cache of spheres maintained by the Sphere Library. Future calls to **sph_draw** and **sph_obj** will re-create and re-cache sphere descriptions as needed.

SEE ALSO

libsphere, sphgnpolys, sphdraw, sphmode, sphobj, sphrotmatrix

NAME

sphgnp – return the number of polygons for the current set of sphere attributes.

FORTRAN 77 SPECIFICATION

integer*4 function sphgnp()

PARAMETERS

none

FUNCTION RETURN VALUE

The return value is the number of polygons in current sphere.

DESCRIPTION

sphgnp returns the number of polygons: triangles, quads, meshed-tris, or stripped-quads, in the current sphere for the set of sphere attributes currently selected by calls to **sphmod.** If the current primitive is not **SPHPOL** or **SPHMSH**, then the number returned is that which corresponds to the primitve **SPHPOL.**

SEE ALSO

libsphere, sphdra, sphfre, sphmod, sphobj, sphrma

NAME

sphgnpolys – return the number of polygons for the current set of sphere attributes.

C SPECIFICATION

int sphgnpolys()

PARAMETERS

none

FUNCTION RETURN VALUE

The return value is the number of polygons in current sphere.

DESCRIPTION

sphgnpolys returns the number of polygons: triangles, quads, meshed-tris, or stripped-quads, in the current sphere for the set of sphere attributes currently selected by calls to **sphmode.** If the current primitive is not **SPH_POLY** or **SPH_MESH,** then the number returned is that which corresponds to the primitve **SPH_POLY.**

SEE ALSO

libsphere, sphdraw, sphfree, sphmode, sphobj, sphrotmatrix

NAME

sphmode – set sphere attributes

C SPECIFICATION

int sphmode(attribute, value)
int attribute, value;

PARAMETERS

attribute
the sphere attribute to set

value
the value to which the attribute is to be set

FUNCTION RETURN VALUE

Returns -1 if an invalid attribute or value is specified, 0 otherwise.

DESCRIPTION

sphmode is used to specify the attributes of a sphere.

The sphere attributes and valid settings are as follows:

SPH_TESS
selects the type of sphere tesselation. Valid settings for tesselation are: **SPH_OCT**, **SPH_ICOS**, **SPH_BARY**, **SPH_CUBE**, and **SPH_BILIN**. These constants correspond to algorithms that tesselate a sphere into polygons based on the following types of tesselation: octahedral, icosahedral, barycentric, cubic, and bilinear tesselation. The octahedral, icosahedral, and cubic methods tesselate by subdividing the original faces of the base form. The barycentric method guarantees that all of the triangles have equal area and has the same number of triangles as the octahedral tesselation. The bilinear tesselation is the traditional longitude-latitude method. The default tesselation is **SPH_OCT**.

SPH_DEPTH
sets depth of sphere tesselation. The number of polygons in a sphere is a function of the tesselation type, depth, and primitive used. Higher depth values result in more polygons, and hence a smoother looking sphere that takes longer to draw. Valid settings for sphere depth are any value from 1 to **SPH_MAXDEPTH**, inclusive. The default sphere depth is 5.

SPH_PRIM
selects the GL drawing primitive. Valid settings for sphere primitive are: **SPH_POLY**, **SPH_MESH**, **SPH_LINE**, and **SPH_POINT**. When the primitive attribute is set to **SPH_POLY**, independent triangles or quads are drawn, depending on the sphere tesselation type. When using **SPH_MESH**, tmeshes or qstrips are

used. The default value for the sphere primitive is **SPH_MESH.**

SPH_HEMI
toggles the hemisphere attribute. When TRUE, less than the entire sphere may be drawn, but all of the non-negative Z portion of the sphere from its canonical orientation will be drawn. Use of this attribute can improve performance when using the **SPH_ORIENT** attribute. The default value is FALSE, meaning that full spheres are drawn.

SPH_ORIENT
to toggle the fixed-orientation attribute. When TRUE, spheres will be displayed in a fixed orientation. This means that when viewing rotations are done, the center of a sphere is transformed to its correct position, and the sphere surface is drawn in a fixed orientation. The default orientation is the canonical orientation unless specified by user with the **sphrotmatrix** cmd. Use of this attribute can reduce some lighting artifacts. When the value of SPH_ORIENT is TRUE, the Sphere Library assumes uniform scaling. When doing non-uniform scaling in viewing transformations the results are undefined. The default value is FALSE so that spheres can be manipulated freely by all matrix operations.

SEE ALSO
libsphere, sphdraw, sphfree, sphgnpolys, sphobj, sphrotmatrix

NAME

sphobj – create a GL object containing a sphere

C SPECIFICATION

void sphobj(objid)
Object objid;

PARAMETERS

objid the integer object identifier of the requested sphere

FUNCTION RETURN VALUE

none

DESCRIPTION

sphobj is used to create a GL object with the requested object identifier, containing a sphere specified by the current sphere attributes set by **sphmode**.

SEE ALSO

libsphere, sphfree, sphgnpolys, sphmode, sphrotmatrix, sphdraw

NAME

sphrma – orient spheres

FORTRAN 77 SPECIFICATION

subroutine sphrma(mat)
real mat(4,4)

PARAMETERS

mat the rotation matrix to specify the orientation of the sphere

DESCRIPTION

sphrma specifies the rotation matrix **sphdra** uses for displaying spheres with a fixed orientation when the sphere attribute, **SPHORI**, is TRUE. The orientation of the sphere is applied by the Sphere Library before any scaling for radius or translation to the location sent to **sphdra**. The matrix provided must be a pure rotation matrix or the results will be unpredictable. When **SPHORI** is TRUE, the Sphere Library assumes uniform scaling; the effects of nonuniform scaling are undefined.

SEE ALSO

libsphere, sphmod, sphobj, sphdra, sphgnp, sphfre

NOTE

sphrma can be used to reduce lighting effects when rotating objects that contain spheres. It can also be used on a VGX, in combination with the accumulation buffer, to antialias spheres. For performance, when **SPHORI** is TRUE, use the **SPHHEM** attribute to draw only the front-facing half of the sphere.

NAME

sphrotmatrix – orient spheres

C SPECIFICATION

void sphrotmatrix(mat)
Matrix mat;

PARAMETERS

mat the rotation matrix to specify the orientation of the sphere

DESCRIPTION

sphrotmatrix specifies the rotation matrix **sphdraw** uses for displaying spheres with a fixed orientation when the sphere attribute, **SPH_ORIENT**, is TRUE. The orientation of the sphere is applied by the Sphere Library before any scaling for radius or translation to the location sent to **sphdraw**. The matrix provided must be a pure rotation matrix or the results will be unpredictable. When **SPH_ORIENT** is TRUE, the Sphere Library assumes uniform scaling; the effects of nonuniform scaling are undefined.

SEE ALSO

libsphere, sphmode, sphobj, sphdraw, sphgnpolys, sphfree

NOTE

sphrotmatrix can be used to reduce lighting effects when rotating objects that contain spheres. It can also be used on a VGX, in combination with the accumulation buffer, to antialias spheres. For performance, when **SPH_ORIENT** is TRUE, use the **SPH_HEMI** attribute to draw only the front-facing half of the sphere.