Release Notice: DG/UX™ System for AViiON® Computers

Release 5.4

Part number 085-600265-01

This release notice applies to the following models:

P001
Q001

Printed in the United States of America
January 1992

Restrictions and Trademarks

DATA GENERAL CORPORATION
4400 Computer Drive
Westboro, Massachusetts 01580

Release Notice:
DG/UX™ System for AViiON® Computers
Release 5.4

085-600265-01

January 1992

Contents

Index

1 Introduction

This release notice describes the DG/UX™ Release 5.4 operating system for Data General's family of AViiON® computers. In addition, this notice supplements information in the current product documentation. It includes information developed after the current manuals were printed and manual corrections.

1.1 Printing This Release Notice

You can print additional copies of this release notice after you have installed the DG/UX System. You can find a copy suitable for lineprinters in the file **/usr/release/dgux_5.4.rn**. If there are differences between the printed copy of this notice and the on-line version, the printed copy takes precedence.

2 Product Description

DG/UX Release 5.4 for AViiON systems is an advanced UNIX® operating system for Data General's line of industry-standard computers. The DG/UX System is a redesigned and reimplemented version of UNIX providing compliance with a wide variety of prominent industry standards.

Data General has added value to the UNIX kernel and file system without compromising adherence to existing UNIX standards, and continues to track other standards including POSIX, FIPS, X/Open, 88open, and de facto standards. The DG/UX System delivers advanced symmetric multiprocessor support with fully concurrent I/O to deliver tangible benefits of multiprocessor technology. Reliability, data integrity, high availability, ease-of-use, and productivity features are delivered in the DG/UX operating system.

The DG/UX software is bundled on the same tape with other products. The products on the tape may include the following:

- DG/UX kernel, commands and libraries
- GNU C compiler(s)
- TCP/IP communications software
- ONC/NFS software
- X Windows software (included only in package P001)
- Looking Glass desktop manager (included only in package P001)
- OSF/Motif extension to X Windows (included only in package P001)
- AView graphics library (included only in package P001)

- On-line manual pages for all products

Following are some of the features and benefits of the DG/UX operating system:

2.1 Standards Compliance

The DG/UX System complies with the following industry and de facto standards:

- AT&T System V Release 4 interfaces
- AT&T System V Release 3 interfaces
- 4.2/4.3 Berkeley Software Distribution interfaces
- POSIX.1 (IEEE Standard 1003.1-1988, FIPS 151-1)
- X/Open XPG3-Base
- 88open Binary Compatibility Standard 1.1A
- 88open Object Compatibility Standard 1.1A
- 88open Binary Compatibility Standard Networking Supplement
- 88open Application Installation Standard
- TCP/IP protocols
- ONC/NFS 4.0 protocols and interfaces
- X Window System Version 11 Release 4
- OSF/Motif 1.1
- COFF object format
- ELF object format
- System V Interface Definitions (SVID) Issue 3 (verified with SVVS)
- C Language (ANSI X3.159-1989)
- IEEE floating-point conformance (IEEE Standard 754-1985)
- SunOS 4.03 tape format compatibility for installation, operation, and use of diskless clients and servers in mixed SunOS and DG/UX environments

2.2 Data Integrity and High Availability

Release 5.4 of the DG/UX System provides the following data integrity and high availability features:

- Intelligent, selective fast recovery file system to provide very rapid recovery of the file system after a crash
- Auto-reboot after power and soft failures to provide higher system availability
- On-line controller reloads and restarts to provide rapid recovery of system resources
- Software disk mirroring to provide higher data availability and protection against single point failures
- Optional High-Availability Disk Array subsystems to provide support for on-line replacement of a failed disk and on-line rebuild of data
- Optional Dual-Ported High Availability Disk Array to support transfer of control from the primary CPU to a standby system in the event the primary fails
- Unbuffered I/O support and "write-verify" options to enhance application and data integrity via rapid and confirmed transfer of data to stable storage media

2.3 Enhanced Diagnostics Capabilities

Release 5.4 of the DG/UX System provides the following enhanced diagnostics capabilities:

- Error logging for disk, tape, and memory
- Optional machine-initiated diagnostic callout through the SMART (System Maintenance and Repair Technology) package

2.4 Ease of Use

Release 5.4 of the DG/UX System provides the following ease of use features:

- Preloaded disks for selected hardware platforms
- The ability to automatically configure a kernel and size peripherals at installation time
- An advanced, simple-to-use operator interface
- The ability to grow and shrink file systems in place without backup/restore operations and without taking the system down
- Very large file systems (up to 2 terabytes) that can span multiple physical disks
- Load-and-go support for binary executables adhering to 88open standards

- Multiple build and execution environments

- The ability to remotely manage mounted file systems, backups, diskless systems, and X terminals

2.5 Performance Enhancements

Release 5.4 of the DG/UX System provides the following performance enhancements:

- Symmetric multiprocessor kernel with fully concurrent I/O to optimize throughput

- Intelligent scheduler tuned for interactive applications to provide sustained throughput

- Dynamic file system buffer caching to provide improved performance for most applications

- Software disk striping to provide improved performance for applications performing sequential I/O on large files

2.6 Innovative Kernel Technology

Release 5.4 of the DG/UX System provides the following innovative kernel technology:

- Applications-transparent symmetric multi-processing

- Symmetric multi-processing STREAMS

- Scalability across the AViiON family

- Ease of maintenance, extensibility, and reliability

2.7 Advanced System Administration

Release 5.4 of the DG/UX System contains a new version of Data General's system administration product, **sysadm**(1M). The 4.3x version of the system administration product is called **osysadm**(1M).

The new **sysadm** product has all the capabilities of **osysadm** but it also provides the following new features:

- A more object-oriented approach to system management: you select an object to be managed, then select the operation to perform on the object

- Multiple user interfaces, including a character-terminal interface similar to **osysadm** plus a graphical OSF/Motif-based interface

- A consistent low-level interface to managed objects

- Customizable access permissions on menus and operations enable non-privileged users to access some system management functions

See section 4.3 for more information on **sysadm**.

3 Environment

3.1 Hardware

Release 5.4 of the DG/UX System runs on Data General AViiON series machines with revision E.2 or later of the MC88100 processor and a minimum of 8 megabytes of main memory, at least 322 megabytes of disk storage, a system console (or graphics monitor for workstations), and a tape drive (for software distribution). The 179 megabyte disk is no longer a supported system disk. If you are presently using a 179 megabyte disk, Data General provides a 179 megabyte to 322 megabyte disk upgrade to help ease your transition. Consult your sales representative for complete information on this option and the other disk drives, tape drives, printers, and terminals supported by the DG/UX System.

You may also run release 5.4 of the DG/UX System on a diskless AViiON workstation. This requires a minimum of 8 megabytes of main memory and console or workstation monitor, but does not require any disk or tape drive units. If you plan to be doing work that puts a heavy windowing load on the system, you should have at least 12 megabytes of main memory on your workstation.

3.1.1 SCSI Controller

Depending on the firmware revision of the SCSI controller in AV5000/AV6000 servers, DG/UX may not support SCSI synchronous transfers. This is caused by a controller limitation. Revision 11 firmware for the VME SCSI controller removes this limitation. During system boot of DG/UX 5.4, the controller revision is checked and if it is not Revision 11 or later, the message:

```
  Firmware in SCSI controller "cisc()" is out of date --
  see release notice.
```

is printed on the system console. This message indicates that the current controller firmware does not support synchronous transfers, which improve performance. There is no risk in using firmware revisions prior to revision 11 since asynchronous SCSI transfers continue to work.

3.2 Software

You may load release 5.4 of the DG/UX system on clean (empty) **/** and **/usr** file systems, or you may choose to upgrade a DG/UX Release 4.3x system. For more information on software loading procedures, see *Installing the DG/UX™ System* (093-701087). The previous installation manual, *Installing and Managing the DG/UX™ System* (093-701052), is now obsolete and should not be used to install release 5.4. See section 8 for more information on installation.

CAUTION: There may be certain layered products that either require this release of DG/UX to run or that do not have a version that runs on this release. Please contact your Data General representative for more information, before attempting to use layered products with this release.

4 Enhancements and Changes

This section contains information about the enhancements and changes made to DG/UX in Release 5.4. The section covers the following areas:

DG/UX Kernel
Programming Commands
The sysadm Command
Other Administrative Commands
General Commands
Libraries
Printer Services
Device Services
Multi-National Language Supplement (MNLS)
X/Open Portability Guide, version3 (XPG3)

4.1 DG/UX Kernel

This section describes the enhancements and changes to the DG/UX kernel.

4.1.1 New Hardware Support

The following hardware is supported for the first time in this release of the DG/UX system:

- AV 7000 and AV 8000 quad-processor servers
- AV 530 and AV 4600 systems
- Model 7421 VME single-ended SCSI controller
- Model 7422 VME differential SCSI controller
- Model 7902 High-Availability Disk Array subsystem
- Model 7416 VME Token Ring controller
- Model G 6488-H Danish keyboard
- Model G 6488-O Norwegian keyboard

4.1.2 STREAMS

DG/UX STREAMS have been rewritten to allow transparent symmetric multi-processing within STREAMS while maintaining compatibility with the AT&T System V Release 4 STREAMS. The amount of concurrency for a given module/driver is selected by specifying a concurrency set in the DG/UX master file (**/usr/etc/master.d/dgux**) where the module/driver is declared. Refer to the comments in the DG/UX master file for more details.

sad

The **sad**(7) (STREAMS Administration Driver) has been added to the DG/UX kernel. Sad enables user programs to specify a list of modules that should be pushed automatically whenever a STREAM is first opened on a specified device number.

4.1.3 Pipes and FIFOs

DG/UX Release 5.4 implements pipes and FIFOs within STREAMS. In addition, pipes have been made full-duplex (that is, data may flow in either direction through a pipe). Previously, data could flow through a pipe only in a single direction. A visible effect of pipes and FIFOs being STREAMS based is that flow control may allow more than **PIPE_BUF** bytes to be written before a write blocks. These changes mirror changes in AT&T System V Release 4 and should be transparent to all but a few applications.

4.1.4 STREAMS Based TTY Support

DG/UX Release 5.4 implements asynchronous terminal support as STREAMS-based drivers and modules in the manner of AT&T UNIX System V Release 4. The modules **ldterm**(7) and **ttcompat**(7) provide the standard line discipline processing. The drivers **duart**(7), **syac**(7), and **syscon**(7) are now STREAMS-based drivers that support the built-in serial ports and the intelligent VME async controllers respectively. See the corresponding man pages for more details.

4.1.5 STREAMS Based pty Support

DG/UX Release 5.4 provides a migration path from BSD style pseudoterminals (ptys) to System V Release 4 style STREAMS ptys by retaining certain BSD style pty behavior (with some restrictions) to ease the transition. The retained behavior will be removed in a future release of the DG/UX operating system. For details, see the "New Features for Future Development" section below.

The move to STREAMS-based ptys does require certain application programs to make code changes. These changes are necessary to maintain conformance with System V Release 4.

Differences

There should be no apparent differences in pseudoterminal behavior to the user of STREAMS pseudoterminals. The differences are in the interface seen by the programmer.

The following is a list of differences that pose potential compatibility problems to existing pty client code.

Pty Line Discipline Manipulation

The biggest difference to the programmer is that the state of the line discipline presented by the slave side of the pty can no longer be obtained or manipulated by ioctls from the master side of the pty.

In earlier releases of DG/UX, most line discipline related ioctls were handled by a set of common low level routines accessible from either the slave or master side of the pty. One artifact of the STREAMS implementation is that the line discipline module (**ldterm**(7)) accepts line discipline related ioctls only from the slave side.

Pty Mode Initialization

Behavior previously requested by certain ioctls is now requested by pushing the appropriate STREAMS module. The **TIOCPKT** operation **ioctl** is now replaced by a push of the **dgpckt** STREAMS module. The **TIOCPTYCTL** operation with a requested service level of 3 is replaced by a push of the **dgpty** STREAMS module.

Pty Default Line Discipline Modes

Some of the default line discipline modes have changed in DG/UX Release 5.4 to match the default modes in System V Release 4. This change should not be of great concern to programs, since most pty initialization code explicitly sets the line discipline modes.

New Features for Future Development

The remaining items represent changes made in DG/UX Release 5.4 that anticipate future changes to the DG/UX System.

Pty Mode Manipulation

DG/UX Release 5.4 provides the BSD interface to packet mode by the **dgpckt** STREAMS module. The module provides the interface described in the include file **<sys/_int_pty_ioctl.h>**, which is different from the System V Release 4 packet mode behavior. To use the **dgpckt** module, the calling program must always issue the read system call with at least a 1024 byte buffer.

Support for DG service level 3 is provided by the **dgpty** module. To use the **dgpty** module, the calling program must issue the **read**(2) system call with at least a 1025 byte buffer.

Support for the **dgpckt** and **dgpty** STREAMS modules will be dropped in a future release of the DG/UX operating system.

Pty Names

The naming conventions have been expanded in DG/UX Release 5.4 to support the System V Release 4 style of pty naming. The new nodes are **/dev/ptmx**, a clonable pty master device, and **/dev/pts/***N*, the pty slave nodes.

Support for the DG/UX pty naming convention, **/dev/ttyp***N*, where *N* ranges from 10 upwards, will be dropped in a future release of the DG/UX operating system.

Pty Allocation

You are encouraged to write programs that use the clonable pty master driver for pty allocation instead of looping through the list of pty masters, attempting to open each until an open succeeds. The new code is very straightforward and more efficient, because it requires only one open:

```
{
           int master;
           int slave;
           int rc;

           char *slavename;

           master = open("/dev/ptmx", O_RDWR);
           if (master < 0)
           {
               perror("master open");
               exit(1);
           }

           slavename = ptsname(master);
           if (NULL == slavename)
           {
               fprintf(stderr, "NULL slavename\n");
               exit(1);
           }

           unlockpt(master);

           slave = open(slavename, O_RDWR);
           if (slave < 0)
           {
               perror("slave open");
               exit(1);
           }
           rc = ioctl(slave, I_PUSH, "ptem");
           if (rc < 0)
           {
               perror("push ptem");
               exit(1);
           }
```

```
            /* push ldterm and ttcompact ... */

            /* program continues ... */
    }
```

NOTE: When you use the clonable ptys, you must push the line discipline modules **ptem**(7), **ldterm(7)**, and **ttcompact** (in that order) onto the slave side of the pty.

Programs that must conform to the BCS Networking Supplement must still loop through the list of Binary Compatibility Standard, Networking Supplement (BCSNS) pty master names, attempting to open each until an open succeeds or the list is exhausted.

4.1.6 System Call Enhancements and Changes

The following system calls were added or enhanced in DG/UX Release 5.4:

dg_paging_info	**dg_set_cpd_limits**	**dg_sys_info**
dg_sysctl	**fchdir**	**fstatvfs**
getcontext	**getpmsg**	**getsid**
mincore	**mmap**	**mprotect**
munmap	**putpmsg**	**reboot**
setcontext	**sigaction**	**sigaltstack**
sigsendset	**statvfs**	**sync**
sysfs	**uadmin**	**waitid**

See below for more information on the **dg_set_cpd_limits**(2), **dg_sysctl**(2), **sigaction**(2), and **sync**(2) system calls. Refer to the individual man pages for information on the other system calls.

dg_set_cpd_limits(2)

Previously, the **dg_set_cpd_limits**(2) system call incorrectly allowed the target cpd's limits to be changed when the caller specified a pathname that had "." or ".." as its last component and the caller did not have write access to the parent directory of the cpd. Now the system call returns **EINVAL** in these cases of "." or ".." as the last component; the limits of the cpd are not changed. Use an absolute pathname instead.

dg_sysctl(2)

A new system call, **dg_sysctl**(2), lets you control several new features related to system panics and reboots. These new features can help to make your systems more highly available by automatically starting a system dump and/or automatically rebooting the system after a panic occurs. You can use the new **dg_sysctl**(1M) command from the shell to control these features.

The new features are:

- You can specify the system behavior for starting a system dump when a panic occurs. You can prompt the operator to start a dump (the default), you can have the dump automatically started, or you can skip the dump.

Currently, the only dump devices supported are tape drives and network devices.

If you choose to have the system dump automatically started, you must ensure that the dump medium is always available and ready to be used. For tape drives, this means that you must have a write-enabled tape in the drive at all times.

If a panic occurs and the dump device cannot be opened (for example, no tape in the drive), the dump is skipped. As long as the dump starts and completes successfully with the available medium, no operator intervention is required. However, if there are any further problems with the dump (for example, hard error on the tape, new tape volume required for a multi-volume dump), the operator is prompted to mount a new tape and respond once the tape is ready.

- You can specify the system behavior after panic processing is complete. You can have the system halted (the default) or automatically rebooted after a panic.

- You can change the device to be used for system dumps.

  This is useful if there are multiple tape drives on the system and the one you normally use for system dumps is broken or being used for some other purpose. In these cases, you could change the dump device to another drive and then switch back to the original drive once it has been repaired or becomes available.

  The default dump device is what you specified for the DUMP configuration variable when you built the kernel. However, you should set the default device using **dg_sysctl** rather than when you build the kernel.

- You can change the boot command line that is used to reboot the system, either automatically after a panic as described above, or when you call the **reboot**(2) or **uadmin**(2) system calls appropriately.

  This is useful if you want to change which kernel is being run, but you can wait until the next time the system is rebooted to begin using it. The default boot command line is the line that was used to boot the currently running system. You can also specify that the boot command line stored by the SCM be used to boot the system.

- You can get the current settings for the auto-reboot and dump start states, for the current dump device name, and for the current boot path command line.

  Only the superuser may set the auto-reboot state, dump start state, dump device name, and boot path command line. Any user, however, may examine these variables.

For diskless clients, a system dump done to device **inen()** is handled differently than system dumps to other devices. The **inen** dump handler always automatically reboots the system with a boot path of **inen() −d**. This path instructs the bootstrap to do the system dump across the network to the server. This means that the autoreboot state is ignored when a system dump is taken to device **inen()**. Additionally, since the kernel is always booted with a boot path of **inen() -d** after such a system dump, any boot path that has been set through **dg_sysctl**(2) is ignored, and **inen() -d** becomes the default boot path for the new kernel. The consequences of this are that all future automatic reboots of the system also cause a system dump to be done, and this may overwrite the previous system dump done by the client. If no panic dumps are done by the client, the autoreboot state is honored.

For more details, see the man pages for the **dg_sysctl**(2) system call, the **/usr/include/sys/dg_sysctl.h** include file, and the **dg_sysctl**(1M) command. Also, see the man pages for the **reboot**(2) and **uadmin**(2) system calls, and the **reboot**(1M) command.

All of the functionality of the **dg_devctl**(2) system call has been moved to the new **dg_sysctl**(2) system call. The **dg_devctl**(2) system call and the **/usr/include/sys/dg_devctl.h** include file are still available, providing both source and binary compatibility with any existing programs that might use them.

The **dg_devctl**(2) system call and **/usr/include/sys/dg_devctl.h** include file will be removed in a future release of the DG/UX System, so any new programs you write should use the new **dg_sysctl**(2) system call and **/usr/include/sys/dg_sysctl.h** include file instead.

sigaction(2)

The **sigaction**(1) call has been enhanced to allow it to emulate the BSD **signal()** and **sigvec()** semantics, as well as the AT&T **signal()** and **sigset()** semantics. You can change signal delivery behavior with the **sigaction()** call by setting (or not setting) particular flags in the **sa_flags** field of the **sigaction** structure. See the **sigaction**(2) man page for an explanation of the valid flags and how each can modify signal delivery.

sync(2)

Previously, there was no way to register a sync routine for a custom built file system, and therefore the file system would not be synchronized when the **sync(2)** system call was executed. The interface to the **fs_register_file_system** kernel routine has been changed to accept pointers to a file system's fs_vfs_operations_type structure which contains a pointer to its sync routine and its fs_vfm_operations_type structure. The **sync**(2) system call now calls the sync routine for each registered file system.

Additionally, the fs_vfs_operations_type structure has been expanded to contain a pointer to the file system's **mount** function. Pointers to the file system's **mount** function and **fh_to_vnode** function used to be passed directly to the **fs_register_file_system** routine.

These changes are important only to people who build customized file system software. The file system types shipped with DG/UX are already modified to use the new interface.

4.1.7 Process Management

Load average calculation

The load average calculation has been modified to reflect only jobs that would run if they were given to a processor. Jobs that cannot run because they are awaiting an event are no longer included in the load average calculation. The new method better reflects system load and is more consistent with the way other UNIX systems report load average.

For more information about the load average, see the **ruptime**(1C) man page, the **dg_sys_info**(2) man page, and the **/usr/include/sys/dg_sys_info.h** include file.

4.1.8 MS-DOS File System

DG/UX Release 5.4 supports direct access to MS-DOS file systems on floppy diskettes. DOS file systems can be mounted like normal file systems and the files accessed directly for reading and writing. See the **mount**(1M) and **dfm**(4) man pages for details.

You can now format floppy diskettes as DOS diskettes using options on the **mkfs**(1M) command.

4.1.9 Terminal Services

The DG/UX Release 5.4 terminal services are significantly different from previous releases. See the sections below on the Service Access Facility and the **getty**(1M), **uugetty**, and **ttymon**(1M) commands for details.

You should also be aware of the following changes:

- DG/UX Release 5.4 and later releases require a slightly different configuration of the terminal controller boards than previous releases. The hardware setup depends on the model of terminal controller in the system. For 16 line MUX controllers and the 128 line controller, switch 1 position 2 must be set to the "on" setting. For the 255 line controller, jumper E22 pins 3-4 must be in. This has always been the Data General recommended setting for these boards, although the DG/UX Release 4.3x driver would work with either setting.

- Support has been added for BCSNS-style pseudoterminals (**/dev/tty**[*p-za-o*][*0-9a-f*]) and System V Release 4 style slave side pseudoterminals (**/dev/pts/***%d*). DG/UX style ptys (**/dev/ttyp***%d*) will NOT be supported in a future release of the DG/UX system. Applications are encouraged to migrate to using clonable (System V Release 4) ptys or, if they are BCS applications, BCSNS pty names.

Service Access Facility

The Service Access Facility (SAF) provides general procedures for service access, so that login access on the local system and network access to certain local services are managed in essentially similar ways. The controlling process (daemon) of SAF is the Service Access Controller (**sac**(1M)). It starts, stops, oversees, and maintains status information on services, which consist of "port monitor" processes running under sac. The most important port monitor is **ttymon**, whose relationship to **getty** and **uugetty** is described in the next section.

You can administer SAF completely through command interfaces. Direct editing of system files used by SAF is highly discouraged, and in some cases will have unexpected effects. Note that the **sysadm** Terminal submenu can provide shortcuts for managing simple terminals. The **sysadm** menus that handle TTY devices are also completely different from the previous **getty** menus.

Administrators should examine the manual *Managing the DG/UX™ System* (093-701088) and the following man pages for more information about SAF and its administrative interfaces:

sysadm(1M)	manage **SAF**, using the ports submenu of the devices menu
admportmonitor(1M)	manage port monitors
admportservice(1M)	manage services for individual ports under a port monitor
admterminal(1M)	manage simple terminal devices
sac(1M)	service access controller
sacadm(1M)	administer port monitors under sac
pmadm(1M)	administer specific port services under a port monitor
ttymon(1M)	port monitor for terminal ports
ttyadm(1M)	format and output ttymon-specific information
sttydefs(1M)	maintain line settings for TTY ports
stty(1M)	set the options for a terminal

getty, uugetty, and ttymon

The **getty**(1M) and **uugetty** commands have been replaced by the **ttymon**(1M) command. This command consolidates their functions and moves their administration into the SAF architecture.

The **ttymon** port monitor manages the login access functions that were formerly handled by **getty**. **Ttymon** differs from **getty** in the following important ways:

- **Ttymon** provides any service the system administrator configures. **Getty** provided only login service.

- Each invocation of **ttymon** can monitor multiple TTY ports. **Getty** supported only one port per invocation.

- **Ttymon** is a persistent process that continues to run after the service process is initiated. The **getty** process was replaced by the process the service invoked.

- **Ttymon** can take advantage of STREAMS I/O capabilities.

- **Ttymon** can configure line disciplines on a per-port basis.

- **Ttymon** provides an optional message display to indicate when a port is disabled from logging in.

A one-to-one replacement of **getty** by **ttymon** is possible. In fact, for backwards compatibility, a link to **getty** is provided. When invoked by this link, **ttymon** accepts the same command line that **getty** did. However, greater flexibility, more features, **sysadm** support, and lower process counts are possible by migrating to use **ttymon** directly.

When you load DG/UX Release 5.4 on top of a previous DG/UX installation, this migration is done automaticany tor **getty** and **uugetty** in **/etc/inittab**. This conversion is done when the system is first brought out of single user mode (during the installation package setup stage).

The **/etc/gettydefs** file has been replaced by the new file, **/etc/ttydefs**. The new **ttydefs**(4) format is more powerful and extensible than the **gettydefs**(4) format. Its most notable features are that line discipline modes are set using the same syntax as the **stty**(1) command, and that it can (and should) be maintained through the **sttydefs**(1M) command instead of by direct editing of the file. Direct editing of the file can cause unexpected results. The conversion process that updates **/etc/inittab** also translates a system's **/etc/gettydefs** file into an equivalent **/etc/ttydefs** file.

The only difference that typical users should notice is the lack of numerous **getty** processes when they issue the **ps -e** command.

4.1.10 Device Drivers and I/O Devices

The following device drivers have been added:

- The **fe** pseudo-device driver has been added. This driver is included automatically in every system and is accessed via the character special device **/dev/fe**.

  It is used by the Data General Field Engineering SMART (System Maintenance And Repair Technology) package and should not be accessed by general applications. If you purchase the SMART package, Field Engineering is able to remotely monitor any system problems you might experience through this driver. If you do not purchase SMART, the driver has no effect on your system. The SMART support offering is optional and can be purchased as a supplement to other service agreements.

- To support the High Availability Disk Array (HADA), two drivers have been added: the HADA adapter driver, **hada**(7), and the disk array device driver, **da**(7). The **hada** is an adapter driver, which allows both **da**(7) and **st**(7) devices to be configured on it.

  Specification of **da**(7) devices follows the nested notation convention used for DG/UX SCSI device/adapter configurations, for example, **da(hada(0),6)**. See the **da**(7) and **hada**(7) man pages for more information.

  Note that all **da**(7) devices must be associated with a unit number, using the **gridman**(1M) utility, before you can use them. See "Binding Drive Modules as Physical Disks" in *Operating the High Availability Disk Array Subsystem* (014-002059-00) and the **gridman**(1M) man page.

  The HADA adapter provides a general-purpose SCSI interface, intended to support tape devices. The SCSI ID for these devices can range from 0 to 5, inclusive, as SCSI IDs 6 and 7 are reserved by the HADA controller hardware. Specification of these devices follows the convention used for other SCSI adapters, for example, **st(hada(),4)**. The SCSI IDs for these general-purpose SCSI devices are permanently associated with unit numbers 0 to 5, inclusive, and require no preparation by **gridman**(1M).

  Note that one of the disks in the HADA may now be the system disk.

- The **dgen**(7) driver has been added. This is the STREAMS driver for the integrated ethernet interface on AV530, and AV4600/4620 systems. The entry "dgen(0)" represents the ethernet interface on the system board. The entry "dgen(1)" represents the secondary ethernet interface that comes on the available expansion I/O card. Diskless booting is supported over the dgen(0) interface only.

- The **ssid**(7) driver has been added. This is the STREAMS Synchronous Interface Driver. It provides the interface to the VSC synchronous controller boards for the DG/UX X.25

product.

- The **vitr**(7) driver has been added. This is the STREAMS driver for the VTC 4/16 Token-Ring controller. It provides access to IEEE 802.5 Token Ring Lan based networks from AViiON class machines.

- The **llc**(6P) driver has been added. It supports IEEE 802.2 Logical Link Control (LLC) connections over IEEE 802.3 CSMA/CD Lans and IEEE 802.5 Token Ring Lans. The DG/UX LLC layer is implemented as a STREAM many-to-many mulitplexor (Pseudo device driver). The layer provides support for multiple protocol modules above it (i.e. SNAP, SNA PC, NetBEUI, Novell, OSI/P, etc.) supporting multiple Token Ring or Ethernet drivers below it. The DG/UX LLC layer provides IEEE 802.2 Class I and Class II classes. This allows support for both Type 1 connectionless service and Type 2 connection oriented service. Type 3 acknowledged connectionless service is not implemented.

- The SNAP STREAMS pseudo-driver implements the IEEE 802.1d standard for SubNetwork Access Protocol. This allows non-802 based transport and network levels (such as TCP/IP) to run over over 802-based media (such as IEEE 802.5 Token Ring Lans).

- In DG/UX 5.4, the local area network STREAMS drivers support the DLPI (Data Link Provider Interface) version 1.3 as defined by AT&T. UNIX International has taken DLPI and made extensions to the AT&T version. Future revisions of DG/UX will match the DLPI standard as it evolves. The drivers that support this interface include **inen**, **hken**, **dgen**, **vitr**, and **llc**.

4.1.11 Disk Management Enhancements

Disk management has been enhanced in the following ways:

- The **diskman**(1M) utility has been significantly enhanced to improve the system installation process and to support automatic detection of the system hardware configuration. For details, see *Installing the DG/UX™ System* (093-701087).

- **Diskman** also has an option for invoking the **gridman** utility to manage the High-Availability Disk Array (HADA). For details on **gridman**, see *Operating the High-Availability Disk-Array Subsystem* (014-002059) and the **gridman**(1M) man page.

- The **diskman** utility has been enhanced to allow the selective growing and shrinking of unmounted file systems. See the section 4.1.12 for more information.

4.1.12 File System Enhancements

The file system enhancements in DG/UX Release 5.4 provide support for large file systems, logical disk mirroring, disk striping, growing and shrinking of unmounted file systems, and fast-recovery file systems. You can use disk mirroring to increase data availability by replicating logical disks. You can use disk striping to improve load balancing across multiple physical devices. A large file system is one containing up to 2 terabytes of data. Previously, the maximum size of a file system was two gigabytes.

See *Managing the DG/UX™ System* (093-701088) for more information.

Logical Disk Mirroring

The operating system provides logical disk mirroring and requires no special hardware support. The High Availability Disk Array (HADA) subsystem provides disk mirroring through the hardware. See the manuals provided with these arrays for details. Logical disk mirroring is available on all AViiON hardware platforms with at least one disk drive. The system administrator can optimize a system's disk availability by creating disk mirrors for important logical disk images, for example, those which must be highly available, on a one-by-one basis. Disk mirroring provides protection against single-point failures. Each image of a disk mirror should have a unique path (separate disk controllers and housings) for maximum benefit.

The data duplication provided by a mirror gives you higher data availability than a single logical disk image. In the event of a disk block becoming unreadable on one component disk image, the mirror transparently reads the equivalent block on another image and repairs the faulty image. Under typical operation, a mirror is composed of two or three logical disk images. A mirror can consist of a single logical disk image, such as when the mirror is being built, when it is being re-formed after a system crash, or when the system administrator has removed the other images for whatever reason. However, such a mirror does not provide increased data availability.

I/O errors that occur when reading from a disk mirror are automatically repaired by the system in a transparent fashion. If the error is unrepairable, the failing logical disk image is removed from the disk mirror and a message is sent to the system error logger.

Disk Striping

Disk striping is the interleaving of sequential physical blocks across multiple devices to improve access rates to the data on the disks. The HADA subsystem provides disk striping through the hardware. See the manuals provided with those arrays for details. Logical disk striping is available on all AViiON hardware platforms with a least 2 disk drives. Disk striping, when used correctly, can result in higher sequential disk I/O performance. This is because readahead done by the file system can be concurrent with normal I/O requests. Also, disk striping may aid performance by balancing I/O requests among physical disk drives and controllers.

Striping is a very powerful mechanism, but it can be misused. For example, you must ensure that the stripe size matches the data element size of files within the file system. Unless the file system data structures are aligned with the stripe size, the striping mechanism does not work correctly. There is a degradation of performance in the unaligned case, because more I/Os would be necessary to retrieve the same amount of data as when not striping.

When using striping, use the default of 16 blocks for both the data element size and the stripe size for the most balanced performance. If you use a stripe size other than 16 blocks, **mkfs**(1M) attempts to change the default data element size to match that stripe size. Smaller stripe sizes perform well in some cases and poorly in others. Using a value larger than 16 blocks sets the default data element size to a large value, which would waste disk space in most cases. A stripe size larger than 32 blocks is not recommended, because the largest buffer can be 32 blocks.

Growing and Shrinking File Systems

Logical disks without a file system may also be grown and shrunk, providing that the logical disk is not currently in use (that is, not mounted). The area to be expanded or reduced is always located at the end of the logical disk.

You can grow a file system by adding new logical disk pieces, or by having the last piece extended (the last piece is automatically extended if the new logical disk piece is contiguous to it on the same physical disk). Shrinking a file system may cause logical disk pieces to be removed from the logical disk and may also cause the last piece to be reduced in size.

All system and user data is compacted into a shrunken file system. You may optionally specify the percent number of free blocks and free file node slots to attempt to leave in each of the remaining disk allocation regions (DARs) when the shrink operation is complete. If the percentage cannot be upheld, the shrink fails (the file system is not modified in this event). Using percentages of zero guarantee that the shrink tries to compact the file system as much as possible. It is possible that the user data and the hidden system data may not fit in the requested smaller file system size.

fsck

DG/UX Release 5.4 introduces the fast-recovery file system. The fast-recovery file system is designed to reduce the length of time it takes to run **fsck** on a file system.

To mount a file system in fast-recovery mode, use the **−o fsck_log_size** option. For example:

```
mount -o fsck_log_size=8 /dev/dsk/foo /foo
```

The number specified in the option is the number of disk blocks to use for the fast-recovery log. You should use a power of 2 between 4 and 32. In **/etc/fstab**, this option is part of the ***opts*** field. Note that there are no spaces between options or around the commas that separate the options.

```
/dev/dsk/foo     /foo     dg/ux    rw,fsck_log_size=8 w 1
```

There is a new flag for **fsck**:

−l Recover in fast-recovery mode if the file system was last mounted in that mode. Recover normally, otherwise. Note that even if your file system has been mounted in fast-recovery mode, normal recovery mode may be used on it by simply omitting this flag.

If you want to use fast-recovery **fsck** on a file that is checked as part of coming to **init** level 3, you should change the declaration of **fsck_ARG** in **/etc/dgux.params** from **-xp** to **-xpl**. If you want the root file system to be fast-recovery, you must specify the **ROOTLOGSIZE** and **FSCKFLAGS** in your system configuration file before building the kernel. For example, set **ROOTLOGSIZE** to **8** and **FSCKFLAGS** to "**-xpl**" in the system file. Be sure to include the quotation marks around the flag.

mkfs

Previously, **mkfs**(1M) could not build a file system with a large number of file nodes relative to the number of data blocks. This prevented building file systems that are to be used to store a large number of very small files. You can now specify any value for the **-i** switch, which indicates the ratio between the number of bytes per file and the number of inodes. However, making file systems with a small **-i** value can produce a file system with more file nodes than can be used, and may take a long time to build.

4.1.13 Subsystem Enhancements and Changes

dev (device drivers) Subsystem

- A write verify **ioctl** has been added to the **sd**(7) disk driver code to support user level write verify operations. The command **dkctl**(1M) provides the user level interface. See the **dkctl**(1M) manual page for more information.

- The lineprinter driver has changed the behavior of the open logic. Previously, if the printer was not ready, the open routing would return with an error. Now the driver loops every five seconds to check if the device has become ready. The open routing continues looping until either the device becomes ready or a signal is received.

- Manual dual porting of **da**(7) disk units is now supported. DG/UX support of this feature allows for two separate AViiON systems, each with its own HADA controller, to share mutually exclusive **da**(7) units of a HADA cabinet.

  You can use manual dual porting to provide a scheme for manual failover to a warm standby machine. For example, system A might register **da** units 6 and 7, while system B registers units 8 and 9. If system A should fail, the HADA controller in system A must be reset using the SCM reset command or the system can be powered off. Once this is done:

  — System B is booted

  — System B then registers **da** units 6, and 7 using **diskman**(1M), in addition to units 8, and 9

  — **Fsck**(1M), or another appropriate check and repair utility, is used to repair any file systems

  — File systems on **da** units 6, and 7 are mounted

  — Operation continues on system B

  When system A is repaired, system B can unmount and deregister **da**(7) units 6, and 7. System A can then be rebooted, can register and mount these units, and continue operation.

  For administrative simplicity, each system should have a unique **system**(4) file which explicitly defines which **da**(7) units are configured and registered at system boot time. During a failover, **da**(7) units can be dynamically registered or deregistered using the physical disk management facilities of **diskman**(1M), then mounted or unmounted.

When **da**(7) units are failed over to the standby machine, the controller begins an operation to verify the integrity of the unit. While the verify operation is in progress, the I/O performance of the controller is degraded, and the unit is vulnerable to a single point of failure. Contact your Data General representative for more information on failover, if you need this function.

io (input/output) Subsystem

- The interrupt initialization sequence has been modified to distribute interrupts across all job processors (JPs) in multi-JP configurations, rather than always interrupting the initial processor. Each interrupt is assigned to a JP at configuration time and remains with that JP for the life of the system.

- The value of **FOPEN_MAX** in **/usr/include/stdio.h** can now be either 20 or 64. Under **-ansi** the value changes to 20.

sfm (STREAMS file manager) Subsystem

- The System V Release 4 versions of the **timod** and **tirdwr** STREAMS modules have been added in this release. System V Release 3 backward compatibility has also been incorporated into the System V Release 4 version of **timod**. This is tied to changes in **<sys/timod.h>**. Code that uses this header file probably needs to be recompiled. This also creates a rev-lock with the TLI library.

sys (/usr/include/sys header files) Subsystem

- MAXIOCBSZ and MAXBSIZE have been moved from under #ifdef _KERNEL to make them visible outside the kernel. This change is for compatibility with AT&T System V Release 4.

- All occurrences of "KERNEL" in the **<sys>** header files have been changed to "_KERNEL" for ANSI C compliance.

- Previously, the **dg_process_info.h** include file incorrectly described the "resident_process_size" field as being the process's resident size in blocks. The explanation has been changed from "blocks" to "pages". The value reported in the field has not changed, only its explanation.

ts (terminal services) Subsystem

- Support for the **KBD_GET_STATE** function of **ioctl** has been added. This function returns a structure to the caller containing the current state of the keyboard LEDs and a count of the scan codes currently in the keyboard driver's internal buffer. With this information you can synchronize your application with the current state of the keyboard.

- The system driver has been changed to allow **syscon**(7) redirection to be overridden by succeeding **TIOCCONS ioctl** calls. The **syscon** driver has been changed to send output to both the console and the redirect device when console redirection is in effect. This prevents the console from appearing to be hung.

4.2 Programming Commands

This section describes the enhancements and changes in software development tools and the SDE (Software Development Environment).

4.2.1 gcc

The **gcc**(1) man page documents compiler options that are unique to Version 2 of GNU C, which is not yet released for general use. DG/UX Release 5.4 provides for the installation and use of multiple versions of GNU C on your system. When Version 2 becomes available, you will be able to add it to your system and use it selectively as dictated by your development requirements.

The new command **default-gcc**(1) is used to query or set the default version of the GNU C compiler when more than one version is available on the system. Both **gcc**(1) and **cc**(1) support options to select a version other than the default.

4.2.2 Assembler Warning Messages

The assembler now displays warning messages when it encounters instructions that might exhibit different behavior on future 88000 processors. Code generated by the GNU C compiler does not produce such warnings. However, these warnings may appear when you use **gcc**(1) or **cc**(1) to translate assembler code from some other source. You should modify any assembler code as required to eliminate these warnings.

Code produced by the current releases of Green Hills compilers, 1.8.5.4, yields a number of the following assembler warning messages:

```
Warning: Double argument starts in odd register
```

These instructions will not fail on future processors, but will carry a performance penalty. The 1.8.6 release of the Green Hills compilers will not generate these instructions.

To suppress these assembler warnings, you may pass the option **−M,88100** to **as**(1). For example:

```
# ghcc -Wa,-M,88100 foo.c ↵
```

4.2.3 The sde Mechanism

The **sde** mechanism now supports the **m88kdguxelf** target environment. This mechanism generates ELF (Executable and Linking Format) objects and executables and supports the production and use of shared objects. This is the default target environment on DG/UX Release 5.4.

See the **sde**(5), **sde-target**(1), and **sdetab**(4) man pages for more information.

4.2.4 The Elink Mechanism

A number of the software development tools (for example, **ld**, **as**) need to find target specific versions of libraries and other files. They do this by incorporating a mechanism called "elink" (Environment variable sensitive symbolic link).

An elink makes non-standard use of the standard symbolic link mechanism to encode a pathname that includes the names of one or more environment variables. When a file is opened, the environment variable is queried to complete the pathname. If the variable is not defined, you may supply a default value to be used instead.

This elink mechanism is unique to the software development tools. Other programs which attempt to open an elink receive an error indicating that the file does not exist.

See the **elink**(5) man page for more information.

4.2.5 Changing SDE_TARGET to TARGET_BINARY_INTERFACE

The current environment variable **SDE_TARGET** is too general a name for the multiple development environment architecture provided with release 5.4 of the DG/UX system. Therefore, **SDE_TARGET** has been replaced with a new environment variable called **TARGET_BINARY_INTERFACE**. Changing the environment variable does not change the **sde-target**(1) command.

4.2.6 cc, as, ld

Cc(1), **as**(1), **ld**(1), and other program development commands now support options unique to the ELF target environment, principally for the generation and manipulation of shared objects. For more information, see the respective man pages for these commands.

Many of the DG/UX System libraries are now dynamically linked by default under release 5.4. You should use dynamic linking, because effective sharing of libraries among different programs reduces the system-wide demand for main memory resources. This reduces paging activity on systems where demand for main memory outstrips supply. These benefits are particularly noticeable for programs whose file sizes drop significantly when they are dynamically linked, such as X clients.

Note, however, that some functions are slower for dynamically linked programs than for statically linked programs. System performance is likely to benefit if some heavily used programs are statically linked. If a program is invoked by **exec**(2) very frequently (at least every few seconds) or calls **fork**(2) very frequently, then you should consider static linking.

4.2.7 ld and ld-coff Man Pages

The man page for **ld**(1) now documents the link editor in the ELF target environment. You can access the man page for **ld** in the COFF target environment as **ld-coff**(1).

4.2.8 The dbx Debugger

The **dbx** debugger has a number of enhancements that are highlighted below. See the **dbx**(1) man page for more information.

- The **dbx** commands **delete**, **catch**, and **ignore** now accept multiple arguments.
- **Dbx** does not ignore a quit command encountered in a **.dbxinit** file.
- The initial display mode in **dbx** is now hexadecimal, not octal.
- **Dbx**'s nomenclature for machine registers is simplified and extended.
- The **alias** command in **dbx** is enhanced to accept an arbitrary one-line character sequence, which may denote arguments as #1, #2, and so on.
- The "trace variable" capability provides greatly improved performance.

4.2.9 Misalignment Exception Handler

The exception handler that repairs misaligned access faults is revised for this release. See the **misalign**(5) man page for more information.

4.3 The sysadm Command

The **sysadm**(1M) command has been completely redesigned to provide a multi-level, object-oriented management framework. At the bottom level are over 40 low-level non-interactive administrative commands, called **adm** commands. At the top are the two user interfaces. In the middle is a program which reads interface-independent menu descriptions, translates them into a form that the interface drivers can use, and invokes the **adm** commands to perform the actual system management.

This design has the following advantages:

- The system is data-driven, allowing you to add new menus and operations (for example, third-party or optional packages) without recompiling.
- The menu names, operation names, query prompts, help messages, and so on are identical in the character interface and the OSF/Motif interface.
- The **adm** commands provide a means for knowledgeable users to

- — bypass the menu hierarchy entirely,
- — create scripts which combine several management operations, and
- — arrange for certain administrative tasks to be performed automatically (for example, using **admbackup**(1M) for automatically performing backups via **cron**(1M)).

The new **sysadm** command automatically determines whether to invoke the line-oriented ASCII interface or the graphical OSF/Motif interface. However, you can specify which interface you want by invoking **asysadm** for the ASCII interface or **xsysadm** for the OSF/Motif interface. See the **sysadm**(1M) man page for more information.

The previous **sysadm** command is maintained in the **osysadm**(1M) command. Many of the operations present in the previous version of **sysadm** are no longer available in **osysadm**.

4.3.1 Adm Commands

Below is a list of the **adm** commands. Each command is in **/usr/bin** and has a Chapter 1 man page entry.

adm88package
admaccounting
admalias
admbackup
admclient
admdate
admdefault
admdumpcycle
admdumpdevice
admether
admfilesystem
admfsinfo
admgroup
admhost
admipinterface
admkernel
admlock
admnetwork
admnls
admpackage
admportmonitor
admportservice
admprocess
admrelease
admresolve
admroute
admrshell
admsar
admservice
admsnmpcommunity
admsnmpobject
admsnmptrap
admsvcorder
admswap
admtape
admtcpipdaemon
admtcpipparams
admterminal
admtrustedhost
admuser
admxterminal

4.3.2 Menus

The **sysadm** menus are arranged differently from the **osysadm** menus. The **osysadm** commands (**addclient** and **setuppackage**, for example) are maintained in **sysadm**. Use **sysadm −l** for a list of menu shortcuts.

The main **asysadm** menu contains the following items:

```
                    Main Menu

1  Session ->            Manage this sysadm session
2  File System ->        Manage file systems
3  System ->             Manage DG/UX system databases
4  Client ->             Manage OS and X terminal clients
5  Device ->             Manage devices and device queues
6  Networking ->         Manage network databases
7  User ->               Manage users and groups
```

```
    8  Software ->           Manage software packages
    9  Help ->               Get help on sysadm and its queries

  Enter a number, a name, ? or <number>? for help,
  or q to quit:
```

The "->" symbol following a menu item name indicates that the item is a menu. Menu items with "..." following the name are operations for which you must supply certain information. Menu item names with no suffix are operations which do not require any additional information.

See the man page for the interface description interpreter, **idi**(1), for a complete description of how to navigate the menus in each of the interfaces. The man page also contains a list of X Window System resources which you may modify to customize the appearance of the OSF/Motif interface.

Below is a list of several frequently-used commands from **osysadm** with the corresponding operation in **asysadm**:

loadpackage	Software -> Package -> Load
setuppackage	Software -> Package -> Set up
newdgux	System -> Kernel -> Build Or you can autoconfigure a custom kernel with System -> Kernel -> Auto Configure.
addclient	Client -> OS Client -> Add
makesrv	(No longer needed)
clientdefaults	Client -> OS Client -> Defaults -> Create
datetime	System -> Date -> Set
addhost	Networking -> TCP/IP -> Hosts -> Add
adduser	User -> Login Account -> Add
fsdump	File System -> Backup -> Create
filerestore	File System -> Backup -> Restore

Terminal and Printer Setup

More information about setting up terminals and printers with the new **sysadm** and examples of the procedures involved in these processes is in *Customizing the DG/UX™ System* (093-701101). See Chapter 5 for setting up terminals and Chapter 6 for setting up printers.

4.3.3 Queries

For operations that require additional information, **sysadm** presents some type of query. Defaults are provided whenever possible. You may encounter the following types of queries:

Text query	Allows entry of arbitrary text.
Selection query	Allows selection of one or more choices, and possibly allows arbitrary text.
Boolean query	Allows a "yes" or "no" response.
Range query	Allows entry of a number from a specified range.

To get Help for queries, enter "?" in the line-oriented interface, or press Function Key 1 in the OSF/Motif interface.

You may answer queries for an operation in any order. Use "^" to return to the previous query in the line-oriented interface. Select the "OK" button in the OSF/Motif interface when you have answered the queries.

See the **idi**(1) man page for a complete description of queries.

For additional information on using **sysadm** and its menus, see *Managing the DG/UX™ System* (093-701088).

4.4 Other Administrative Commands

Many of the administrative commands have been upgraded to include System V Release 4 content. Individual commands which are new to DG/UX or whose interfaces have changed are covered below. Refer to the command's man page for more information.

4.4.1 autocon

The **autocon** program is run by **init**(1M) as a **sysinit** entry in the **inittab**(4) file. It initializes STREAMS line discipline modules on the system console device.

4.4.2 chk.fsck

The **chk.fsck** script now mounts local file systems before exiting. This ensures that file systems are mounted before **chk.system** runs, because it (and others) may need some of these file systems (that is, the file systems under **/srv**).

This script now provides parallelism in **fsck** checking and recovers the external cache data.

4.4.3 chk.strtty

The **chk.strtty** script is a new script that arranges for automatic pushing of STREAMS modules for TTY devices. The set of TTY devices initialized is controlled by the **strtty_ARG** variable in **/etc/dgux.params**.

4.4.4 chk.system

The **chk.system** script now provides support for **dkctl**(1M), **setuname**(1M), **dg_sysctl**(1M), and automatic operator notification of reboots using the **reboot_notify_START** parameter. This parameter is found in **/etc/dgux.params**.

4.4.5 config

The **config**(1M) command handles a new master file format that allows dynamically assigned major device numbers and different levels of STREAMS concurrency.

4.4.6 cron

For DG/UX Release 5.4, some of the files associated with **cron**(1M) have moved to new locations. Prior to release 5.4, all **cron** files were located in the directory **/var/spool/cron.** The directories associated with **cron** and their contents for DG/UX Release 5.4 are as follows:

/etc/cron.d:	**cron.allow** **cron.deny** **at.allow** **at.deny** **queuedefs**
/var/cron:	**log**
/var/spool/cron:	**atjobs/** **crontabs/**
/etc/default:	**cron**

Note that the new file **/etc/default/cron** is used to set configurable options for **cron**. See the **cron(1M)** man page for more information.

4.4.7 devnm

The **devnm**(1M) command now prepends "/dev" to the mount device name to correspond to SVID-3. This affects scripts that might use this command.

4.4.8 df

The **df**(1M) command now supports the **-g** option in compliance with SVID. The **-g** option overrides other options. The **-o** and **-V** options are also supported.

The new **-c** flag instructs **df** to use Control Point Directory (CPD) limits when calculating statistics.

4.4.9 dg_kill

This is a new command. It is a superset of **sh**'s built-in **kill**. The **dg_kill**(1) command accepts command names in addition to process ids. You can use this feature to signal processes without using **ps** pipelines to determine the pid of the process.

4.4.10 dumptab

All **dumptab**(4) devices now have a default blocking factor of 16. Prior to DG/UX 5.4, the devices used a blocking factor of 10.

New entries for WORM devices, and entries for "cartridge" and "reel", have also been added.

4.4.11 dump2

The **dump2**(1M) command now supports inclusion and exclusion lists. The **−I** *filename* and **−E** *filename* options specify the name of a file to read for inode numbers that will be included or excluded from the dump. Note that these lists are used in addition to the date from **/etc/dumpdates**. By default, all inodes are in the include list and no inodes are in the exclude list. This change involves a slight degradation of performance during initialization.

Several bugs have been fixed, including one which caused a SIGSEGV in an incorrect **semctl**(2) call. Also fixed is a bug which affects directories whose size is almost a multiple of 512. The size of these directories is now determined correctly.

4.4.12 init

Support has been added to **init(1M)** for new run levels: 'i', 5, and 6. Run level 'i' is intended for use as an installation-specific run level. Run level 5 brings the system down to the firmware monitor. Run level 6 brings the system down and reboots it. See the **init**(1M) man page for more information.

4.4.13 inittab

Lines have been added to the **inittab**(4) script for **autocon** and for **chk.strtty**. These commands are described in this section. Lines have also been added for **sac**, which is the controlling process for the Service Access Facility.

4.4.14 installman

This is a new command for managing system installation. The command presents a series of steps that may be necessary for completing system installation, including setting up packages and building and booting a custom kernel.

See the **installman**(1M) man page for a more information.

4.4.15 oampkg

The 88open Consortium has adopted the AT&T System V Release 4 package installation standard. These commands include **pkgadd**(1M), **pkgask**(1M), **pkgchk**(1), **pkginfo**(1), and **pkgrm**(1M), for use by those installing packages. Additional commands, **pkgmk**(1), **pkgparam**(1), **pkgproto**(1), and **pkgtrans**(1), may be used by package developers to create packages.

The oampkg family of commands uses two device tables that are managed by the **devmgmt** family of commands. These tables are **/etc/device.tab** and **/etc/dgroup.tab**. **/etc/device.tab** contains attributes of devices and aliases for devices. For example, qtape0 is the alias name for the character-special device **/dev/rmt/0**. An attribute of qtape0 is "bufsize=8192," that is, the buffer size a **devmgmt** command is to use when accessing this device is 8 kilobytes. **/etc/dgroup.tab** contains groupings by category of the devices found in **/etc/device.tab**. The **devmgmt** commands that manipulate these tables are **devattr**(1M), **devfree**(1M), **devreserv**(1M), **getdev**(1M), **putdev**(1M), **getdgrp**(1M), **listdgrp**(1M), and **putdgrp**(1M).

4.4.16 oamuser

The **oamuser** family of commands have been added.

The **useradd**(1M), **usermod**(1M), and **userdel**(1M) commands manage the password database, **/etc/passwd**. They enable the superuser to add, modify, and delete users without directly editing the password database.

The **groupadd**(1M), **groupmod**(1M), and **groupdel**(1M) commands manage the group database, **/etc/group**. They enable the superuser to add, modify, and delete groups and members of groups without directly editing the group database.

4.4.17 osysadm

The **osysadm**(1M) command is provided for upward compatibility of product administration scripts that have not been upgraded to the release 5.4 **sysadm**. **Osysadm** should not be used to administer system features for which **sysadm** has available functions. Not all of the operations contained in the old **sysadm** work in **osysadm**. You should not use this command to manage DG/UX 5.4 releases.

NOTE: This command will not be supported in future releases.

4.4.18 probedev

The **probedev**(1M) command is a new command. It is used by the autoconfigure operation of **admkernel**(1M) to automatically construct a custom **system** file before building a kernel. The **probedev** command determines which hardware devices are attached to the system. The devices which **probedev** finds are written to standard output in DG/UX common format.

4.4.19 ps

The **ps**(1) command now supports the **-j**, **-c**, and **-s** options as specified in the SVID.

4.4.20 rc scripts

These scripts now use the **dg_kill** command instead of **ps** pipelines.

The output from these scripts has been greatly streamlined. The verbose output from these scripts is logged to the **/etc/log/init.log** file.

4.4.21 rc.cron

The **rc.cron** script has been modified to reflect the new location of the FIFO file (**/etc/cron.d** instead of **/var/spool/cron**).

4.4.22 rc.init

The **rc.init** script now supports the 'i' installation-specific run level.

4.4.23 rc.links

The **rc.links** script now allows an optional entry for run level 'i' in linktab files. If the value for 'i' is not given, the default is "-".

Rc.links is now implemented in the C programming language.

4.4.24 rc.update

The **biod** startup script has been added to **rc.update**. **Biod** startup used to be part of the NFS rc scripts.

4.4.25 reboot

The **reboot**(1M) command is new in DG/UX Release 5.4. It enables the system to be halted and rebooted. See the man page for a complete description of this command.

4.4.26 restore

The **restore**(1M) command has been enhanced to allow the user to enter "q" at the "Specify next volume #: " prompt.

4.4.27 sar

The **sar**(1) system activity report utility has been enhanced and changed in the following ways:

- The values reported by **sar -b** now reflect only the buffering of system data. In previous releases, these values also reflected the buffering of file data.

- The **rclm/s** value reported by **sar -p** now includes reclaims of file data buffers for the free list.

  The **pgfil/s** value reported by **sar -p** has the same meaning under 5.4 as previous releases, but its values tend to be lower. This is because program files are now paged according to their optimum transfer size, rather than the system page size (4 KB) used in previous releases. The optimum transfer size for a local file is equal to the data element size, within an allowable range of 4 KB to 16 KB. The optimum transfer size for a remote file accessed via NFS is 8 KB when the default **mount** options are used.

- The meaning of the **freemem** value reported by **sar -r** has been changed. Under 4.3x, this value reported only the number of pages on the system's free list. Under 5.4, the report also counts the pages that are not free, but that appear to be unused (unbound) at present.

4.4.28 timezone

The files **/etc/TIMEZONE** and **/etc/TIMEZONE.csh** now include values for the **NLSPATH** and **LANG** variables. See the **timezone**(4) man page for more information.

4.4.29 valtools

Several new validation tools, **valtools**, have been added to validate input from a user. They may be used interactively (the commands that begin with "ck") or to validate information that has already been obtained from the user (the commands that begin with "val"). See the **valtools**(1) man page for more information.

4.4.30 xdrtoc

The **xdrtoc**(8) command now supports the "tarZ" file type. The "tarZ" format is for compressed **tar** images.

4.5 General Commands

Many commands have been added or changed in DG/UX Release 5.4. They have been upgraded to System V Release 4 and in many cases include changes needed to support the international features of the Multi-National Language Supplement (MNLS) and X/Open Portability Guide (XPG3). The XPG3 Compliance Package released with DG/UX Release 4.32 is no longer required. DG/UX Release 5.4 incorporates its contents. More details on MNLS and XPG3 are available below.

The following lists show the changed and added commands. For more information. see the appropriate man pages.

4.5.1 Changed Commands

The following commands have been changed:

Command	Why
awk	SVR4 & MNLS enhancements
bnu commands	SVR4 enhancements
calendar	SVR4 enhancements
cancel	SVR4 enhancements
cat	SVR4 & MNLS enhancements
cb	VR4 & MNLS enhancements
cd	SVR4 enhancements
chgrp	SVR4 enhancements
chmod	SVR4 enhancements
chown	SVR4 enhancements
cp	SVR4 enhancements
cpio	XPG3 enhancements
csh	SVR4 & XPG3 enhancements
csplit	SVR4 enhancements
cu	SVR4 & MNLS enhancements
cut	MNLS enhancements
date	SVR4 & XPG3 enhancements
diff	SVR4 enhancements
diff3	SVR4 enhancements
echo	SVR4 enhancements
ed	SVR4 enhancements
egrep	SVR4 enhancements
expr	SVR4 & XPG3 enhancements
fgrep	SVR4 & MNLS enhancements
file	SVR4 enhancements
find	SVR4 enhancements
grep	SVR4 & XPG3 enhancements
id	SVR4 enhancements
join	MNLS enhancements
kill	SVR4 enhancements

labelit	SVR4 enhancements
lex	SVR4 enhancements
locale	XPG3 enhancements
ln	SVR4 enhancements
lp subsystem	SVR4 enhancements
ls	SVR4, XPG3 & MNLS enhancements
mail	SVR4 & MNLS enhancements
mailx	SVR4 & MNLS enhancements
mknod	SVR4 enhancements
mv	SVR4 enhancements
newform	MNLS enhancements
nl	SVR4 enhancements
od	SVR4 enhancements
paste	MNLS enhancements
pg	SVR4 & MNLS enhancements
pr	SVR4, XPG3 & MNLS enhancements
prof	SVR4 enhancements
rm	SVR4 enhancements
sccs commands	SVR4 enhancements
sdiff	MNLS enhancements
sed	SVR4 enhancements
sh	SVR4 enhancements
sort	SVR4 & MNLS enhancements
split	SVR4 enhancements
stty	SVR4 enhancements
tail	SVR4 enhancements
tar	XPG3 enhancements
tr	MNLS enhancements
uniq	SVR4 & MNLS enhancements
vc	SVR4 enhancements
vi	SVR4 enhancements
wall	SVR4 & MNLS enhancements
wc	SVR4 enhancements
who	XPG3 enhancements
write	SVR4 & MNLS enhancements

4.5.2 Added Commands

The following commands have been added:

Command	Why
alpq	new MNLS
atq	new SVR4
atrm	new SVR4
autopush	new SVR4
chrclass	new XPG3
compress	new SVR4
colltbl	new XPG3
cscope	new SVR4
devattr	new 88open pkg. inst.
eucset	new MNLS
exterr	new XPG3
finger	new SVR4
fmt	new SVR4
fmtmsg	new SVR4
gencat	new XPG3
gettxt	new SVR4
iconv	new MNLS
installf	new SVR4
jsh	new SVR4
kbdcomp	new MNLS
kbdload	new MNLS
kbdpipe	new MNLS
kbdset	new MNLS
ksh	new SVR4
listusers	new SVR4
logins	new SVR4
lpfilter	new SVR4
lpforms	new SVR4
lpsystem	new SVR4
lpusers	new SVR4
mkfifo	new XPG3, SVR4
mkmsgs	new XPG3
montbl	new XPG3
nlsadmin	new SVR4
passmgmt	new SVR4
pmadm	new SVR4
printf	new SVR4
rdump	new
removef	new SVR4
sac	new SVR4
sacadm	new SVR4
setuname	new SVR4
slink	new SVR4
spline	new
srchtxt	new MNLS
strchg	new SVR4
strconf	new SVR4

sttydefs	new SVR4
ttyadm	new SVR4
ufsdump	new SVR4 (symbolic link to dump)
ufsrestore	new SVR4 (symbolic link to restore)
uncompress	new SVR4
wchrtbl	new MNLS
zcat	new SVR4

4.5.3 Static versus Shared Commands

There are two types of libraries in DG/UX: statically-linked and dynamically-linked. A statically-linked library, or archive, is a collection of object files. When you use a function in this type of library, a copy of the object file that contains the function is incorporated in your executable at link time. A dynamically-linked library, or shared object, is a single object file that contains the code for every function in the library. When you call one of these library functions in your program, and specify a dynamic linking option on the **cc** command line, the entire contents of the shared object are mapped into your process's virtual address space at run time. So a shared object contains code that can be used simultaneously by multiple programs at run time. This means the compiled programs are smaller.

A command is referred to as a shared command if it has been built with at least one dynamically-linked library. Most of the commands in this release of DG/UX are shared commands. The exceptions are listed here.

The following **/sbin** commands are built static (that is, using only statically-linked libraries):

autocon
sh
init
umount
halt
su
reboot
fsck
mount
chk.fsck
shutdown

The following **/usr/bin** commands are also built static:

cat
mv
dump2label
rm
cpio
restore
ls
cp
login
chmod
pwd
ln

4.5.4 stty

The **stty**(1) command has been updated to the unified source of AT&T System V Release 4. The command line syntax of the **berk_stty** command has been retained. For backwards compatibility, links are provided to **att_stty** and **berk_stty**. However, there is no longer any functional difference among these three names. The merged STREAMS line discipline in DG/UX Release 5.4 prompted this unification. This new line discipline combines the features of (and supersedes) the previous DG/UX, AT&T, and Berkeley line disciplines.

4.5.5 tar

The **tar**(1) command has been upgraded to be compatible with System V Release 4. In addition, the following enhancements have been made:

- Several changes have been made to allow reading DG/UX Release 4.3x **tar** images that do not have type information in the mode field.
- Trailing "/" characters are stripped from file names given on the command line.
- Root-relative files are now extracted correctly.
- The verbose output format is similar to the DG/UX Release 4.3x (BSD-based) format.
- Control point directories are supported.
- The 't' output includes the major and minor numbers for devices and "fifo" for FIFOs.
- Modification times for directories are set. The System V Release 4 version of **tar** does not reset directory modification times.
- The **setlocale()**(3C) and **cftime()**(3C) functions are used for displaying the date and time fields of the verbose table of contents.
- Archives created on standard input or read from standard output are now handled correctly. If the file is "−", the blocking factor is set to NBLOCK. NBLOCK is now 32, instead of 20. This has no impact on previous archives.

4.5.6 Code Set Conversion and att_kbd

The new DG/UX **att_kbd** STREAMS module is named differently from the **kbd** STREAMS module provided by AT&T. The name change in DG/UX was required to avoid a name conflict with the DG/UX keyboard driver **kbd**. Most documentation has been changed to the new name. Any remaining references to "kbd" should read "att_kbd".

The name change is not expected to have any effect on development procedures or portability. The **att_kbd** module is used with STREAMS TTY by pushing the module with the **strchg**(1) command.

In the AT&T standard environment, the following command is executed:

```
# strchg -h kbd ↵
```

In the DG/UX environment, the following command is executed:

```
# strchg -h att_kbd ↵
```

Once the module is pushed, the module name is no longer required by any action. In addition, no **kbd*** related commands deal with the module name.

A list of the available code set mapping tables can be found in **iconv**(1). In addition, mappings between 8859-1 and DGI (the Data General international code set) are available in types b, d, e, and p. However, the spelling of the code set names used by **iconv** may be slightly different from that used by **att_kbd**. To determine the exact spelling required by **att_kbd**, look at the names of the files in **/usr/lib/kbd**.

The programs **iconv** and **kbd_pipe** provide similar functionality (converting a file from one code set to another). **Iconv** generally gives better performance.

The **att_kbd** "compose key" table for 8859-1 (**/usr/lib/kbd/88591.cpz**) enables you to enter 8859-1 character codes from keyboards that do not have keys for those characters. The compose key is ^T (control-T). To enter a control-T, you must type control-T twice. To enter any of the other characters listed here, you must type control-T followed by one of the character pair(s) listed under "Keys Pressed". Usually you can reverse the two keys you must press. For example, if "ab" is listed, "ba" works as well. Alternative key sequences are listed for some of the characters. "<space>" means you should press the space bar.

Character	Keys Pressed
non-break space	<space><space>
quotation mark	"<space>
number sign	++
apostrophe	'<space>
commercial at	AA
opening bracket	((
backslash	// /<
closing bracket	))
circumflex accent	^<space> ><space>
grave accent	‘<space>
opening brace	(-
vertical line	/^ VL vl
closing brace	-)
tilde	~<space>
inverted !	!!
cent sign	c/ C/ c| C|
pound sign	l- L- l= L=
currency sign	xo XO xO Xo x0 X0
yen sign	y- Y- y= Y=
broken vertical bar	!^ || VB vb
section sign	So SO sO so S0 s0 S! s!
diaeresis	""
copyright	cO Co CO co c0 C0
Female Ordinal	a_ A_
left angle quotation mark	<<
logical not	-,
SHY	--
registered trademark	RO
macron	-^ __
Degree sign	0^ ‾

Degree sign	0*
Plus minus	+-
Superscript 2	2^ 2S 2s
Superscript 3	3^ 3S 3s
acute accent	’’
micron sign	u/ U/
paragraph sign	p! P!
middle dot	.^ ..
cedilla	,,
Superscript 1	1^ 1s 1S
Masculine Ordinal	o_ O_
right angle quotation mark	>>
one quarter	14
one half	12
three quarters	34
inverted ?	??
A grave	A‘
A acute	A’
A circumflex	A^ A>
A tilde	A~ A-
A umlaut	A"
A circle	A*
AE dipthong	AE
C cedilla	C,
E grave	E‘
E acute	E’
E circumflex	E^ E>
E umlaut	E"
I grave	I‘
I acute	I’
I circumflex	I^ I>
I umlaut	I"
capital Iclandic eth	D-
N tilde	N~ N-
O grave	O‘
O acute	O’
O circumflex	O^ O>
O tilde	O~ O-
O umlaut	O"
multiply	xx
O slash	O/
U grave	U‘
U acute	U’
U circumflex	U^ U>
U umlaut	U"
Y acute	Y’
capital Iclandic thor	TH
sharp s	ss
a grave	a‘

a acute	a’
a circumflex	aˆ a>
a tilde	a˜ a-
a umlaut	a"
a circle	a*
ae dipthong	ae
c cedilla	c,
e grave	e‘
e acute	e’
e circumflex	eˆ e>
e umlaut	e"
i grave	i‘
i acute	i’
i circumflex	iˆ i>
i umlaut	i"
small Iclandic eth	d-
n tilde	n˜ n-
o grave	o‘
o acute	o’
o circumflex	oˆ o>
o tilde	o˜ o-
o umlaut	o"
divides	-: -;
o slash	o/
u grave	u‘
u acute	u’
u circumflex	uˆ u>
u umlaut	u"
y acute	y’
small Iclandic thor	th
y umlaut	y"

4.5.7 /etc/ascii

The **ascii** file has been moved from **/etc** to **/usr/pub**.

4.5.8 assist

Since **assist** and **astgen** are no longer supported by AT&T in System V Release 4, they are no longer supported in DG/UX Release 5.4.

4.5.9 Documenter's Tool Kit

The Documenter's Tool Kit (DTK) package is no longer included in the base DG/UX System. A separate package, Xroff, is available.

Xroff provides the same capabilities as DTK, plus additional features. Please see your Data General Representative for ordering information.

The DTK packaged with previous releases of the DG/UX System will continue to run on DG/UX Release 5.4. DTK **troff** can be used with **dpost**(1), the PostScript postprocessor supported in Release 5.4.

If you are upgrading your system from DG/UX Release 4.3, DTK is not overwritten and will be available for your use. If you are doing a clean install of release 5.4 and want to have DTK available, be sure to backup DTK before the installation. You can then restore DTK after you install 5.4. A file list for DTK is in **/usr/release/dtk_4.30.fl**.

4.6 Libraries

4.6.1 Support for Alternate Software Development Environments

DG/UX Release 5.4 supplies the broadest support in the industry for alternate Software Development Environments (SDEs). The DG/UX 5.4 SDEs consist of the following:

- 88open BCS
- 88open OCS
- DG/UX COFF
- DG/UX ELF

Programs built using the BCS SDE run on 88K systems that support the BCS standard, including DG/UX systems beginning with Release 4.30. Programs built using the OCS SDE run on 88K systems that support the OCS standard, including DG/UX systems beginning with Release 5.4. Programs built using the DG/UX COFF or DG/UX ELF SDEs run on DG/UX systems beginning with Release 5.4. The DG/UX ELF SDE supported in DG/UX Release 5.4 provides the flexibility necessary to support the System V Release 4 style of shared libraries.

Beginning with DG/UX Release 5.4, Data General plans few additional functional enhancements to the BCS, OCS, and DG/UX COFF SDEs. Functional enhancement to the DG/UX operating system will concentrate on the DG/UX ELF SDE beginning with DG/UX Release 5.4. However, Data General will continue to correct existing problems with the COFF-based environments.

To assist customers in taking advantage of the benefits of the DG/UX ELF SDE, Data General will maintain support for the DG/UX COFF SDE of the DG/UX operating system (Q001 and P001 model numbered products) until at least July 1992. Data General intends to drop the DG/UX COFF SDE at some time thereafter. However, applications built with the DG/UX COFF SDE will continue to run.

88open is working toward a final version of the System V Application Binary Interface (ABI) Motorola 88000 Processor Supplement. Data General intends for the DG/UX operating system (Q001 and P001 model numbered products) to offer complete support for the 88K ABI in a future release. The 88K ABI is expected ultimately to supplant the existing 88open BCS and OCS. To assist customers in taking advantage of both the 88open BCS/OCS and the 88K ABI, Data General will maintain support for the BCS and OCS SDEs until at least July 1993. Data General intends to drop the BCS and OCS SDEs at some time thereafter.

4.6.2 Available Environments

Several new Software Development Environments (SDEs) are provided with DG/UX Release 5.4. Available environments include the following (new environments are marked with an asterisk (*)):

m88kdguxelf* This is the System V Release 4 ELF environment. This is the environment that contains ELF shared libraries. There are also several static libraries.

m88kdguxcoff* This is the System V Release 3 COFF development environment. It is provided for development of COFF-based applications to run on DG/UX Release 5.4 or later. This environment corresponds to **m88kdgux** on 4.3x systems.

m88kocs* This COFF environment provides the 88open OCS standard interface. It should be used for developing applications that include object files and libraries that are intended to be OCS-compliant. OCS certification has not taken place at this time.

m88kbcs This environment is essentially unchanged from 4.3x. It is intended for developing BCS-compliant applications consisting entirely of statically linked COFF executables. It provides a **libc** compatible with the System V Interface Definition, Edition 2, and contains BSD, POSIX, and ANSI extensions.

m88kdgux This environment is a synonym for one of the other environments. It provides the default if no other environment is explicitly requested. In 4.3x, this was a synonym for what is now **m88kdguxcoff**. In DG/UX Release 5.4 this is a synonym for **m88kdguxelf**.

All of the above environments are intended to provide the set of interfaces specified by the OCS standard. In the ELF environments, these interfaces may not appear in the same libraries specified by the OCS. This is due to restructuring by AT&T in System V Release 4. See below for more information. See also the *DG/UX™ Transition from UNIX System V Release 3 to UNIX System V Release 4* whitepaper. This paper is contained in **/usr/release/V3toV4.wp**.

4.6.3 m88kdgux and m88kdguxelf

m88kdgux is now a synonym for **m88kdguxelf**, making the latter the default DG/UX software development environment. There are some differences in the libraries provided for **m88kdguxelf** and **m88kdguxcoff** (the default environment in DG/UX Release 4.3x). These are described below.

Moved Functions

Several functions have been moved to different libraries in this new environment. This was motivated primarily by the need to follow the AT&T System V Release 4 Application Binary Interface (ABI) and reference port source standard.

Function	Previous location	New location
regcmp	libPW	libgen
regex	libPW	libgen
nlist	libc	libelf

In addition to the above list, many functions normally found in **libc** in the **m88kdguxcoff** environment are found in **libdgc** in the DG/UX Release 5.4 ELF environment. This split facilitates future ABI compliance. The general classes that have been moved are the RPC/YP related functions, socket/internet functions, domain name service functions, Berkeley 4.2 and 4.3 extensions, and miscellaneous DG extensions. For a complete list, see *Supplement for Programmer's Guide: ANSI C and Programming Support Tools* (086-000180).

To accommodate makefiles that assume the DG/UX Release 4.3x **libc** function contents, the **cc** command automatically statically links **libdgc**, even if **-ldgc** isn't specified. If **-ldgc** appears on the **cc** command line, the standard search rules are applied first (that is, look for **libdgc.so**, then **libdgc.a**).

New Libraries

The following new libraries are available in the **m88kdguxelf** environment:

Library	purpose
libadm	to build administrative commands
libcmd	to build miscellaneous commands
libdgc	DG specific - formally part of libc
libelf	object format interfaces
libform	forms management
libgenIO	special purpose device routines
libmail	mail file interfaces
libmenu	menu management
libnsl	System V Release 4 ABI networking (TLI based)
libpanel	extended curses window management
libpkg	general purpose application packaging
libw	wide character library (8-bit)
libw32	wide character library (32-bit)

libc

In the **m88kdguxelf** environment, **libc** contains the following enhancements:

New System Calls		
fchdir	**fstatvfs**	**getcontext**
getpgid	**getpmsg**	**getsid**
isastream	**mincore**	**mmap**
mprotect	**munmap**	**nuname**
putpmsg	**setcontext**	**sigaltstack**
sigsendset	**statvfs**	**sysfs**
sysinfo	**uadmin**	**waitid**

New Library Entries		
addseverity	**catclose**	**catgets**
catopen	**endspent**	**fattach**
fdetach	**fgetspent**	**fmtmsg**
fpclass	**getdate**	**getdate_err**
getspent	**getspnam**	**getsubopt**
gettxt	**grantpt**	**isnan**
isnanf	**lckpwdf**	**logb**
memalign	**modff**	**msync**
nextafter	**nftw**	**nl_langinfo**
psiginfo	**psignal**	**ptsname**
putspent	**scalb**	**setspent**
sigsend	**swapcontext**	**tcgetsid**
ulckpwdf	**unlockpt**	**unordered**

Changed or Enhanced Library Entries
ttyname

Enhanced Standard I/O

The System V Release 4 Standard I/O package adapts to the number of per process file descriptors. This is a system defined limit, based on the configuration variable SDESLIM.

New Memory Functions

There are new implementations of the **memcmp()**, **bcopy()**, and **memset()** functions. The functions are significantly faster for some cases. These new memory functions are available from all SDEs.

Changed Functionality

Cuserid() has been changed to give the System V behavior by default in the **m88kdguxelf** environment. To get POSIX behavior (that is, use the effective userid instead of real), specify **-lposix** on your link line.

Fflush() in System V Release 4 correctly sets the file position on read descriptors per POSIX. This means that the **-lposix** switch is not needed to obtain this behavior in the **m88kdguxelf** environment. However, the switch is still needed in all the COFF environments if the POSIX behavior is desired. See section 5.7.3 for more information on **fflush()**.

New Signal Implementation

The header file **sys/signal.h** changes calls to **signal(), sigset(), sigaction()**, and **sigvec()** into calls to **__xsignal(), __xsigset(), __xsigaction()**, and **__xsigvec()** respectively. This translation is required to support object compatibility and should be transparent. However, it makes every source that includes **sys/signal.h** in an ELF environment include references to these four __x functions.

New malloc Implementation

The **libc malloc()** implementation has been changed by AT&T in System V Release 4. It now uses a tree structure, and should be faster.

New Floating Point Exception Defaults

In the COFF environments, the default floating point environment remains the same by enabling the divide by zero, floating point overflow, and invalid operation exceptions. These exceptions would be disabled if **<ieeefp.h>** was included. In the ELF environment, the default floating point environment has changed to be compatible with AT&T and does not enable any of the exceptions. See the **intro**(3) man page for more information.

libdgc

The **libdgc** library contains two new system calls: **dg_paging_info** and **dg_sysctl**.

libPW

Although AT&T has dropped **libPW** from the System V Release 4 reference port, it is still available in COFF format in the **m88kdguxelf** environment. However, applications using **libPW** for regular expressions should switch to **libgen**. The implementation there is more complete.

The **libPW** function **rename()** has been renamed to **pw_rename()**, so it does not conflict with the **rename()** function in the various **libc**'s available in DG/UX.

libm

The math library has been enhanced by AT&T to have greater accuracy over a wider range of input values. Error behavior is now selectable between ANSI and traditional (**matherr**) mode.

To obtain ANSI behavior for math routines, you must specify the **-X***c* switch to **cc**. The values for **c** are as follows:

t (traditional)
call **matherr**, print error message, set **errno**

a (ANSI)
call **matherr**, set **errno**

c (conformance)
set **errno**

The **-X***c* switch places one of three modules from **/usr/lib** on the **ld** command line: **values-Xt.o, values-Xa.o**, or **values-Xc.o**.

If none of the above modules are specified, traditional mode is the default.

The following new functions have been added to **libm**:

acosh **asinh** **atanh**
cbrt **copysign** **lgamma**
remainder **rint**

4.6.4 m88kdguxcoff

The libraries in this environment are essentially the same as in DG/UX Release 4.3x. Profiling of the **mem*()** and **str*()** functions is now available in the profiling version of **libc**.

4.6.5 m88kbcs

The libraries in this environment are essentially the same as in DG/UX Release 4.3x. Profiling of the **mem*()** and **str*()** functions is now available in the profiling version of **libc**.

4.6.6 m88kocs

The libraries in this environment are the same as those in the **m88kbcs** SDE. The difference is at compile time, where an **_M88KOCS_TARGET** macro is automatically defined (if you are using **cc**, see section 4.6.7 below). This macro modifies certain system header files to be OCS compliant.

4.6.7 Header File Use

Many new header files appear in DG/UX Release 5.4. These new header files relate to functionality that is only available in the **m88kdguxelf** environment. There are also several new feature test macros in DG/UX Release 5.4. Feature test macros allow developers to select header file features in specific ways.

Source Environment Macros

The DG/UX Release 5.4 header files support the following source environment macros. These may be used to aid in porting from or writing code that conforms to specific source standards. ANSI C is the minimum source standard and is always implied. Macros marked with an asterisk (*) are new.

_BSD_SOURCE
_DGUX_SOURCE
_M88KBCS_SOURCE
_POSIX_SOURCE
_SYSV3_SOURCE
_SYSV4_SOURCE*
_XOPEN_SOURCE*

Target Environment Macros

The DG/UX Release 5.4 header files support several binary target environments. A binary target is a way of selecting libraries that supply interfaces defined in binary standards. Examples are 88open BCS and OCS. The following macros are automatically turned on by the **cc** command depending on the current setting of the **TARGET_BINARY_INTERFACE** (see the **sde-target**(1) man page) environment variable. New targets are marked with an asterisk (*). Note that there is no **_DGUXELF_TARGET**. This macro is implied by the default environment, **_DGUX_TARGET**.

NOTE: These target macros are NOT guaranteed to be turned on automatically by third party compilers. You must define one of these target macros when using any target environment other than the default.

_DGUXCOFF_TARGET*
_DGUX_TARGET
_M88KBCS_TARGET
_M88KOCS_TARGET*

Source Flavor Macros

The DG/UX Release 5.4 header files support several combinations of source environment features. These combinations are called flavors and are controlled by the following macros. New flavors are indicated with an asterisk (*). See the indicated header files for information on the function of each macro. Note that **_SYSV3_LIMITS_FLAVOR** automatically turns on **_SYSV_LIMITS_FLAVOR**.

Macro	Header Files
_BSD_SIGNAL_FLAVOR	**<sys/signal.h> <setjmp.h>**
_BSD_TTY_FLAVOR	**<sys/_int_bsd_tty_ioctl.h> <sgtty.h>**
_BSD_WAIT_FLAVOR	**<sys/wait.h>**

_POSIX_BAUD_RATE_FLAVOR	**<sys/termios.h> <sys/_int_termio_tty_ioctl.h>**
_SYSV3_BAUD_RATE_FLAVOR	**<sys/termios.h> <sys/_int_termio_tty_ioctl.h>**
_SYSV3_LIMITS_FLAVOR	**<limits.h> <sys/_int_features.h>**
***_SYSV_LIMITS_FLAVOR**	**<limits.h> <sys/_int_features.h> <sys/_int_unistd.h>**

4.6.8 Transport Layer Interface Libraries

In this release there are two versions of the Transport Layer Interface libraries: **libnsl_s.a** and **libnsl.so**

- **Libnsl_s.a** is a COFF-based library that supports the AT&T System V Release 3 TLI interface.

- **Libnsl.so** is an ELF-based shared library that supports the AT&T System V Release 4 TLI interface. This library not only contains the TLI interfaces as in **libnsl_s**, but also includes new functionality for use with the Service Access Facility (SAF), and transport independent network selection at run-time (**netdir** and **netselect**).

The TLI portion of **libnsl.so** has passed the System V Release 4 SVVS tests. The supplied TIMOD accommodates programs linked with either the System V Release 3 TLI from **libnsl_s** or the System V Release 4 TLI form **libnsl**.

4.7 Printer Services

4.7.1 LP Subsystem

The LP printer subsystem is now based on the AT&T System V Release 4 model. Prior to release 5.4, the LP subsystem was based on the Berkley model. The LP printer subsystem has also been changed in the following ways:

- PostScript is officially supported in 5.4. You can install filters to convert input to PostScript for printing on PostScript printers to support enhanced features.

- Printers on a print server may be logically grouped in a class. You may select the class rather than the printer as the destination. The first available printer in the class is chosen.

- You can describe forms, such a checks, through **sysadm**(1M). When a request is sent specifying a form, the request is held in the queue until the administrator puts that form on the printer and records the form as mounted.

- Jobs can be assigned priorities within the bounds administrator sets for that user.

- When printing to a 5.4 or BSD print server from a 5.4 system, the remote print job is available on the job queue for the system on which the print job originated. This allows **lpstat**(1), **cancel**, and **lpmove** to affect remote print jobs without requiring the user to login to the remote system first.

- The **cancel -u** enables the user to cancel multiple requests.

- You may define printers with multiple printer-types or content-types. See the **lpadmin**(1M) man page for more information.

- The directory structure has changed significantly from previous releases of DG/UX. The file formats have remained the same in most cases. **/var/lp/logs**, **/etc/lp/***, **/usr/lib/lp/***, and **/var/spool/lp** are the four major directory structures where you can find printer related files.

- The interface program that initializes the printer, calls filters, prints banners, and sends the files to the printer is run as the uid of the user printing the files. Interface programs of releases prior to DG/UX Release 5.4 were run with the uid **lp**.

  When you use the **remshlp** interface program to a printer on a pre-5.4 print server, you must make sure that the user can **remsh**(1C) to the remote host. You can do this with a .rhosts file on a per user basis, or with an entry in the **/etc/hosts.equiv** of the remote server on a per system basis. You can test this by having the user type **remsh** *remote host* **ls**. You should see the appropriate **ls** output. If you see an error message, then possible problems include the following:

  — Proper entries in the **/etc/hosts.equiv** or .rhosts missing

  — The user does not have a login to the remote system

  — If the user is using the **csh** there, .cshrc could be causing a problem; you could use a .cshrc file of a user without a problem for troubleshooting

See *Managing the DG/UX™ system* (093-701088) for more information.

4.8 Device Services

4.8.1 Enhancements

Device services have been enhanced in the following ways:

- The Extended Terminal Interface (ETI) from System V Release 4 has been added. This programming interface includes three new libraries: **libform.a**, **libmenu.a**, and **libpanel.a**.

- System V Release 4 STREAMS commands have been added as follows: **autopush(1M)**, **strchg**(1), and **strconf**. In addition, the **autopush** user interface has been enhanced to support file names as well as device numbers.

- The **curses** library has been upgraded to System V Release 4. Many of its functions are documented for the first time, although they were present in earlier versions.

- The **termcap** library has been upgraded to 4.3 BSD.

- The following commands have been upgraded to System V Release 4: **captoinfo**(1M), **infocmp**(1M), **tic**(1M), **chgtinfo**(1), and **tput**(1).

- The **man**(1) command has been upgraded to 4.3 BSD compatibility. Most notably, "apropos" and "whatis" features have been added.

4.8.2 Changes

Device services have been changed in the following ways:

- **Curses** header files have been rewritten for compliance with the various source standards supported by the rest of the DG/UX system. The following headers were affected: **<curses.h>**, **<term.h>**, and **<unctrl.h>**.

- The **curses** library has been changed to avoid triggering warnings from 88open Binary Compatibility Standard (BCS) test suites. While the feature that caused these warnings was not in violation of the BCS (it was not invoked on any system except DG/UX, where the behavior was well defined), a way was found to retain the feature while avoiding the warnings.

- The **tput init** command correctly initializes color terminals.

- The **termcap** library correctly adjusts the amount of padding to match the current baud rate.

4.9 Multi-National Language Supplement (MNLS)

DG/UX Release 5.4 includes several important internationalization features found in System V Release 4 and MNLS. The capabilities described in this section enhance DG/UX by permitting the development of applications which make no assumptions about language or code set. This results in more opportunities to develop highly portable applications.

With DG/UX Release 5.4, a platform exists to support AT&T localization products. DG/UX Release 5.4 also provides the following portions of the European Language supplement:

- Locale databases for 89 locales

- PostScript printer files to support printing of ISO character sets

DG/UX 5.4 MNLS 4 support consists of the following features:

- Multibyte Character Support: This is based on the Extended UNIX Code (EUC), which provides a mechanism for supporting up to 4 code sets concurrently (ASCII plus 3 others). The additional codesets may be single or multibyte. The program locale determines which codesets are used. EUC multibyte characters are converted to a wide character type for processing within applications. The EUC multibyte functions exist in two libraries: **libw** (providing multibyte analogues and extensions to C library functions) and **libcurses** (enhanced to support multibyte handling). Various utilities use the multibyte support functions.

- STREAMS based Code Set Conversion and Character Mapping: Provides transparent character mapping for code set mapping and compose/dead key functions. It also allows remapping of the keyboard layout.

4.9.1 MNLS Content for DG/UX Release 5.4

The objective for DG/UX Release 5.4 is to provide complete multibyte handling, but not message support. Therefore, the localized system messages feature of MNLS 4 is not included. The following sections list the specific portions of MNLS 4 that are available in DG/UX Release 5.4.

UNIX Commands

The commands are enhanced to provide multibyte support for files and file names. The commands that are enhanced from System V Release 4 are shown in the table below. The function level indicators are defined as follows:

f reads/writes characters without requiring recognition from either the primary code set 0 or the supplementary code sets (1,2 and 3)

r reads/writes characters with recognition from either the primary code set 0 or the supplementary code sets (1,2, and 3)

m has some multibyte character processing functionality, for example, pattern matching

u does not expect non-ASCII characters because of functional specification

Command Name	Level
cat	f
cb	f
cflow	f
cu	f
lint	f
mail	f
srchtxt	f
wall	f
write	f
ls	r
mailx	r
newform	r
pr	r
sdiff	r
uniq	r
awk	m
cut	m
grep	m
join	m

paste	m
pg	m
sort	m
tr	m
eucset	u
wchrtbl	u

Multibyte Character Processing Library

The **libw** library provides several enhancements to support multibyte characters. This library provides wide character functions not provided in **libc**. The wide character type definition can be 16 bits in **libw** (**libw16**) or 32 bits (**libw32**); only the 32-bit **libw** are provided in DG/UX Release 5.4. Man pages are provided which describe the multibyte handling capabilities. The **libw** library provides support for:

- 16/32 Bit Process Codes: The functions in **libw** process the ANSI C defined data type **wchar_t** to represent EUC multibyte characters as discrete characters with a uniform length (wide characters).

- Character/String Functions: **libw** provides wide character analogues of the basic string and character handling functions found in **libc**.

- Character Class and Conversion Functions: **libw** also contains functions for testing and case-conversion to wide characters.

MNLS Headers

DG/UX Release 5.4 includes the following new header files:

getwidth.h	declares **getwidth()**
libw.h	declares **libw** functions
synonyms.h	contains MNLS **#defines**
wctype.h	contains definitions for international functions
widec.h	contains definitions for international functions
xctype.h	defines international classification functions for multibyte characters

Curses Library

The **libcurses** library has been enhanced to support multibyte character handling. Existing functions are changed and new functions are provided. Only **libcurses** for 32-bit wide characters (**libcurses32**) are provided in DG/UX Release 5.4. Man pages are provided which describe the multibyte handling capabilities.

The new DG/UX **libcurses32** library has an inherent limitation. Some functions can not use the video attributes in **chtype** when the object of that type is used to handle 32 bit process codes.

This limitation should not affect current curses applications because the limitation exists in the **wchar_t** handling libraries, and no application currently uses these library functions.

The following functions cannot use the video attributes:

addwch
addwchnstr
addwchstr
inwch
inwchnstr
inwchstr
mvaddwch
mvaddwchnstr
mvaddwchstr
mvinwch
mvinwchnstr
mvinwchstr
mvwaddwch
mvwaddwchnstr
mvwaddwchstr
mvwinwch
mvwinwchnstr
mvwinwchstr
waddwch
waddwchnstr
waddwchstr
winwch
winwchnstr
winwchstr

Keyboard/Character Mapping

The **att_kbd** STREAMS module performs user-defined keyboard mapping (to/from the internal EUC code set for input/output), string translation, and character composition to generate characters that are not otherwise provided. The following items are provided. See the respective man pages for more information.

Command	Description
alp(7)	STREAMS module for algorithm pool management
alpq(1)	command to query the alp STREAMS module
att_kbd(7)	STREAMS module for generalized string translation
kbdcomp(1M)	command to compile **att_kbd** tables
kbdload(1M)	command to load or link **att_kbd** tables
kbdpipe(1)	use the **att_kbd** module in a pipeline
kbdset(1)	attach to **att_kbd** mapping tables; set modes

4.10 XPG3

This section describes the XPG3 (X/Open Portability Guide, version 3) components for DG/UX. The separate XPG3 Compliance Package that accompanied DG/UX Release 4.32 is no longer required. The contents of that package have been incorporated into DG/UX Release 5.4. These components provide base-level compliance with the X/Open Portability Guide, version 3, and consist of:

- New and updated commands
- Updated libraries
- New and updated on-line manual pages
- New databases of international localization information

XPG3 was available as a separate package starting with DG/UX Release 4.32. In DG/UX Release 5.4, XPG3 functionality is included in the default set of ELF static and shared libraries. It is not available in the BCS/OCS or COFF libraries.

4.10.1 Environment Variables for Internationalization

Several internationalization features are incorporated into various library routines, new and modified commands. These features depend upon certain environment variables being set to know which locale to use. These environment variables fall into two classes: old "DG/UX Release 4.3x" and "new XPG3". The old DG/UX Release 4.3x variables were established before the standard was formed and may be used by some third-party applications. The new variables are those promulgated by XPG3, and newer versions of AT&T System V.

For commands and libraries supplied by DG, the old variables are no longer needed. Their use has been subsumed by the new variables. The old variables are useful only with applications developed outside of DG, and linked with libraries predating DG/UX Release 5.4. If you are not already using the old variables, it is highly unlikely that you will have any need to start using the old variables.

The old DG/UX Release 4.3x variables are **CHRCLASS** and **LANGUAGE.** **CHRCLASS** was used in conjunction with the **setchrclass** subroutine, to determine which data file in the directory **/usr/lib/chrclass** to use for character classification. The new version of **setchrclass** is designed to use the variables **LANG** and **LC_CTYPE.**

The **LANGUAGE** variable was used by some commands and applications to control the presentation of the time and date. For all commands and subroutines supplied by DG, it has been subsumed by the variables **LANG** and **LC_TIME.**

The new variables are **NLSPATH**, **LANG**, and the **LC_** variables. These variables are described in the **setlocale**(3C) man page.

NLSPATH helps determine where the **catopen**() function looks for X/Open message catalogs. When this package is being set up by **sysadm setuppackage**, the setup script adds lines to the files **/etc/TIMEZONE** and **/etc/TIMEZONE.csh** to set the value of **NLSPATH** to **/usr/lib/msg/nls/%L/%N:/etc/nls/msg/%L/%N**. See the **catopen**(3C) man page for more information.

LANG is also used by **catopen** in determining where to look for X/Open message catalogs. In addition, **LANG** is used by **setlocale()** to establish the locale for all the other internationalized functions (see below). The value of LANG should correspond to the name of one of the subdirectories under **/usr/lib/locale**. The possible choices provided with this package are listed in Chapter 4 of *Managing the DG/UX™ System* (093-701088). The setup script shipped with this package arranges for the files **/etc/TIMEZONE** and **/etc/TIMEZONE.csh** to contain lines that set the **LANG** environment variable to **C**. You can modify the lines in these files to suit your locale.

You can use the **LC_** variables, **LC_CTYPE** (character types), **LC_NUMERIC**, **LC_TIME**, **LC_COLLATE**, **LC_MONETARY**, and **LC_MESSAGES** to override the effect of **LANG** for particular internationalization categories. Normally, you do not need to set these variables. Their application is described in the **setlocale**(3C) man page. The first five of these are promulgated by X/Open, and are described in the XPG3. **LC_MESSAGES** is an AT&T extension to this model, and is used by the **gettxt()** routine to determine where to look for AT&T message catalogs. Data General does not recommend using **gettxt()** and related facilities for use in applications that are intended to be portable. All the facilities that have been internationalized are designed so that if the locale is specified as **C**, or is left unspecified, the facilities exhibit their old, preinternationalization behavior. In the U.S.A., you may prefer to specify your locale (that is, set the value of the **LANG** environment variable) to **en_US**. One distinction between **en_US** and **C** is that the **en_US** character classification table is ISO 8859-1, the standard for all of Western Europe, while the character classifications under **C** are strictly ASCII. This means text that includes non-ASCII characters such as an e with an umlaut is treated as a printable character under **en_US**, but as an unprintable character under **C**.

4.10.2 Libraries

As specified by XPG3, the **setlocale()** function establishes the locale for one or more localization categories (**LC_CTYPE**, **LC_NUMERIC**, **LC_TIME**, **LC_COLLATE**, or **LC_MONETARY**). There is a sixth category, **LC_MESSAGES**, that is handled in much the same way, but it is not mentioned in the XPG3. This category was added by AT&T. The values assigned to these categories affect the behavior of various functions. This is covered in the **setlocale**(3C) manual page. In addition, the functions **nl_langinfo()**, **localeconv()**, and **gettxt()** are affected by several of these localization parameters.

Many of the internationalization features, such as message catalog facilities (see **gettxt**(3C)) that are in the **m88kdguxelf** environment are not in the **m88kdguxcoff**, **m88kbcs**, and **m88kocs** environments. Many other internationalization features, such as **strcoll**(3C) are present, but offer only **C** locale behavior.

4.10.3 Commands

The following new commands are included in this package:

Command	Description
colltbl(1M)	compiles collation (sorting order) table
gencat(1)	Compiles X/Open-style message catalogs
mkmsgs(1)	Compiles AT&T-style message catalogs
montbl(1M)	compiles monetary information table

4.10.4 Message Facilities

The facilities for internationalizing text messages break down into two categories: X/Open and AT&T. Both of these categories involve storing messages in message catalogs outside of executable programs. This package includes full support for both of these models, but includes only a few message catalogs, and a few pieces of code that take advantage of the messaging facilities.

The X/Open model of messaging includes the **gencat** program to pre-process message catalogs, and the **catopen()**, **catgets()**, and **catclose()** functions to use messages inside C programs.

The **catopen()** function uses the values of the environment variables **NLSPATH** and **LANG** to locate message catalogs. Message catalogs for DG/UX commands are kept in subdirectories under the directory **/usr/lib/msg/nls**. The name of each subdirectory is the locale it supports. The normal way to establish your locale is to assign the value **/usr/lib/msg/nls/%L/%N** to the environment variable **NLSPATH**, and set the environment variable **LANG** to the name of your locale. The setup script provided with this package, that is run during the package setup stage of installation, adds lines to the files **/etc/TIMEZONE** and **/etc/TIMEZONE.csh** to establish values for these environment variables. You can edit the files to change these values.

The AT&T model of messaging includes the **mkmsgs** program to pre-process message catalogs and the **gettxt()** function to use messages inside C programs. Data General does not recommend using these AT&T facilities in applications that are intended to be portable.

The function **gettxt()** uses the locale value established by **setlocale()** for the localization category **LC_MESSAGES** to determine from which subdirectory of **/usr/lib/locale** to get message catalogs. The catalogs actually have names like **/usr/lib/locale/C/LC_MESSAGES/uxsyserr**. The normal way to establish your locale is to set the environment variable **LANG** to the name of your locale, although you can override the value of **LANG** by giving a value to the environment variable **LC_MESSAGES**.

5 Notes and Warnings

This section describes special aspects of DG/UX Release 5.4. It also describes situations where you might experience a problem using this release of DG/UX. The section covers the following areas:

DG/UX Kernel
Programming Commands
The sysadm Command
Other Administrative Commands
General Commands
Time Zones
Libraries
Printer Services
Device Services

5.1 DG/UX Kernel

5.1.1 Overriding Default Values for File System Data Element Sizes

Do not modify the data element size for directories to be greater than the default value provided by the system. Note that if you are using LDU striping and have a stripe size of 32 or greater, you must override the directory data element size by specifying **-S 4** as an argument to **mkfs** when creating that file system. If you must set your directory data element size to be larger than the default, contact your Data General Support Center for further information and assistance.

5.1.2 Dynamic Buffer Cache

Support for dynamic buffer caching has been added in DG/UX Release 5.4. The observed performance profile of the system will be different from DG/UX Release 4.3x. Due to dynamic buffer caching, many application mixes perform significantly better than they did under 4.3x, while others may perform worse.

In DG/UX Release 4.3x, the buffer cache was restricted to a fixed portion of main memory. In 5.4, the buffer cache can use most of main memory. When a significant portion of main memory is being used by the buffer cache, paging performance may degrade..

5.1.3 Configuration Changes in Release 5.4

STREAMS

Warnings

You can now attach STREAMS to the file system. See the **fattach**(3C) and **fdetach**(3C) manual pages for more information. Under some circumstances a program that tries to **fdetach** an attached pipe that is simultaneously being opened may block the open indefinitely. This problem occurs only on attached pipes containing the **connld** STREAMS module. If this occurs, you must kill the process. To avoid this problem, you should take the following three precautions. First, do not explicitly **fdetach** the end of the pipe containing **connld**. Second, do not push **connld** onto both ends of the same pipe. Third, do not attach both ends of the same pipe containing **connld**. If you follow these guidelines, the kernel automatically performs the detachment when the pipe is closed.

Configuration Changes

STRMSGSZ has been changed. If it is set to zero, the per module/driver **maxpsz** is used. Also, if **STRMSGSZ** is set to zero and the module/driver **maxpsz** is **INFPSZ**, no limit is imposed. If **STRMSGSZ** is greater than zero, it specifies a global limit such that any per module/driver **maxpsz** that is greater than **STRMSGSZ** is ignored in favor of the global **STRMSGSZ** limit.

File System

The **PERCENTBUF** configuration variable is no longer valid. This is due to the implementation of the dynamic buffer cache.

Terminals and Pseudo-Terminals

Make sure that the proper modules and drivers (listed below) for STREAMS-based terminals and pseudo-terminals have been configured in your kernel. Add them to your system file, if they are not already there.

ldterm
ttcompat
ptem
dgpckt
dgpty

ptm()
pts()

- Names of TTYs have changed for workstations that have a mouse. **/dev/tty00** has become **/dev/mouse**. **/dev/tty01** has become **/dev/tty00**, and so forth.

- DUART hardware flow control is now supported by the new hardware control interfaces defined in AT&T System V Release 4. DUART was on by default in DG/UX Release 4.3x, but you must explicitly turn it on in Release 5.4 using the new interfaces.

fast_fsck.log

If you need to file a Software Trouble Report (STR) about the fast-recovery file system, please include **/etc/log/fast_fsck.log** in the report. Since this file is appended to every time fsck runs, you should remove it occasionally. Please also include in the STR all output produced by the fast-recovery **fsck** and the normal **fsck** run that reported problems. When **fsck** is run as part of an **init** script, that output is placed in **/etc/log/fsck.log**. If you're running **fsck** by hand, you should redirect output to a file or pipe it through **tee**.

For more information on filing STRs, see section 9.

5.1.4 Systems Heavily Using STREAMS Resources

A system that is heavily using STREAMS resources may reach the system limit on allocated STREAMS queue pairs. When this limit is reached, the system should display the error SFM_ENOSR_QUEUE_LIMIT. However, the system panics instead.

You can fix this problem by setting the queue pair limit to a higher value. Use the **NQUEUE** variable in the system configuration file (**/usr/etc/master.d/dgux**) to do this. If your system is experiencing these panics, you should set **NQUEUE** to a value greater than 2048 default. If you set **NQUEUE** to an arbitrarily high value (for example, 65536), there is no practical queue pair limit and the system always gets the resources it needs. The only limitation is that the **NQUEUE** value must fit into an unsigned 32 bit integer.

You may have trouble recognizing that your system has paniced due to the queue pair limit being reached. If you anticipate very heavy STREAMS usage, you may want to increase the value of **NQUEUE** as a precaution.

5.1.5 FIONREAD Behavior

The behavior of the FIONREAD command for the **ioctl**(2) system call has changed in DG/UX Release 5.4 for STREAMS-based devices. These devices include terminals, pipes, and FIFO special files. Specifically, the command now returns the number of bytes in the first STREAMS message available for reading. This behavior is consistent with System V Release 4, but differs from 4.3 BSD and the 88open Binary Compatibility Standard Networking Supplement. According to these standards, the FIONREAD command should return the total number of bytes that are available for reading by the next **read**(2) system call.

Very few or no applications will be affected by this change, because the command still correctly reflects whether any data is available. The difference is that the number of bytes reported may be lower than expected. Most applications loop to ensure that they read all available data.

Any application affected by this change should be modified to allow either behavior. This command is likely to revert to the previous behavior in a future release of the DG/UX System.

5.1.6 BCS ptrace Extensions

DG/UX Release 5.4 complies with 88open Binary Compatibility Standard (BCS) 1.1, rather than 88open BCS 1.1A, for the optional extensions to **ptrace**(2). Specifically, a process cannot use **waitpid**(2) to debug an attached process that is not its child. This support will be added in a future release of the DG/UX System.

5.1.7 Name of the Root File System's Logical Disk

When the DG/UX System is brought up, the bootstrap program requires two logical disks. The first is the logical disk containing the root file system and the second is the logical disk used for swap space. The system assumes these disks are named "root" and "swap." If these logical disks are not actually named **root** and **swap**, then you must provide the name of these disks when the system is booted. Use the **-q** option on the boot command line and you are prompted to supply the names of the root and swap logical disks. For example, **b sd(insc( ),0) -q**.

Note that you cannot autoboot the system if you must supply the names of these disks to the bootstrap. If you want to be able to autoboot your system, you must name your root and swap logical disks **root** and **swap**.

5.1.8 Adding New Hardware Devices

DG/UX Release 5.4 introduces the autoconfigured kernel, which requires rebuilding of the kernel whenever a new hardware device is added. However, you can still add new hardware devices without rebuilding the kernel using the **sysadm** "System -> Kernel -> Build" operation. See *Managing the DG/UX™ System* and *Customizing the DG/UX™ System* for more information.

5.1.9 Disk Management

If you are using the **diskman**(1M) utility to shrink a logical disk, be aware that **diskman** makes no assumptions about the format of logical disks that do NOT contain file systems. If you shrink a logical disk that does not contain a file system, the blocks on the disk are simply truncated. You may back out of an errant shrink of a logical disk by growing the logical disk in the exact same place it was before you did the shrink. However, if the disk is written to in the time between the shrink and the grow operations, this recovery does not work.

5.1.10 da Units

When you unbind a **da**(7) unit and then rebind the disk modules that composed that unit into another configuration, the short device names in **/dev/rpdsk** and **/dev/pdsk** can become inconsistent with the **da**(7) long names to which the short names are linked. When you unbind and reconfigure **da**(7) units, delete the short names in **/dev/rpdsk** and **/dev/pdsk** and use the **/usr/sbin/init.d/chk.devlink** utility to recreate them.

5.1.11 Mounting a Local File System

You might experience a problem when you are attempting to mount a local file system. This problem occurs in the following situation:

- The local file system mount fails, AND
- One of the following messages appears on the console:
  - the file system has been converted to read only access, OR
  - the file system has been sealed

When this occurs, run **fsck**. This almost always solves the problem and you are then able to do the mount successfully.

If the mount hangs when you try it again, note that just the mount process is hung, not your system. However, this hung process cannot be killed without rebooting the system. If the mount fails a second time, you should reboot the system from the system console or from some other available console at a convenient time. This enables you to mount the file system successfully.

You could decide to operate temporarily without the file system on which the mount hung, then resolve the problem with a reboot at a later time. If you choose to delay the reboot, note that any attempt to mount or unmount any file system will also hang until you reboot.

5.1.12 Mounting Floppy Diskettes

3.5 and 5.25 inch floppy disks must be hard formatted before you try to mount them. If you try to mount a disk that is not hard formatted, you receive error messages. You can use the **mkfs**(1M) command to hard format floppy disks.

5.1.13 Small Data Element Sizes

In previous releases of DG/UX, you could gain significant performance benefits in some applications by using a data element size of one or two 512 byte blocks for a file. Using a small data element improved performance in applications that buffered small amounts of data in a random fashion. In DG/UX Release 5.4, the minimum buffer size is a hardware page (currently 4096 bytes, or eight 512 byte blocks), so reducing the data element size of the file below eight blocks does not produce the same performance benefits as in previous releases. In fact, using a data element size smaller than eight blocks for a file may result in degraded performance, because the system must do more than one I/O to fill the page from the file.

5.1.14 Booting from Model 7421/7422 VME SCSI Controllers

Version 5.10 of the AV5000/AV6000 PROM is required to boot from model 6677 tape drives on model 7421 and model 7422 VME SCSI Controllers. Older versions of the PROM cause boot operations to hang. Other models of DG tape drivers are not affected by this problem. See your Data General representative to obtain version 5.06 AV5000/AV6000 PROMs.

5.1.15 Mirrored Disk Management

When you build disk mirrors, all images added after the first image are marked corrupt. This is to prevent their use until they are synchronized with a non-corrupt image. Images are also marked corrupt if a write fails on the image, causing it to be inconsistent with other images. You should never select a corrupt image as the master of a synchronization, although you can take this action. Doing this may result in a system panic with panic code 16000010, indicating there are no images available. Instead, you should only supply the corrupt image as a slave image for mirror synchronization.

If an image is corrupt and you want to use it as the master of a synchronization, break the mirror and rebuild it supplying the previously corrupt image as the first image added to the mirror. In this case, the first image is not be considered corrupt and you can use it as the master for synchronizing other images on the mirror.

5.1.16 ETXTBSY Detection

ETXTBSY detection has been removed in DG/UX Release 5.4. Previously, if an executable was running and someone tried to compile a new version of the executable, the compilation failed and the system displayed the error message "Text file busy". In 5.4, the compilation succeeds and the running executable core dumps.

5.1.17 Diskless Clients as Servers

As in previous releases of the DG/UX system, diskless clients do not respond to Reverse Address Resolution Protocol (RARP) requests. Therefore, a diskless client cannot act as a server for another diskless client nor as a boot server for an Xterminal.

5.1.18 System Dumps on Diskless Clients

For diskless clients, a system dump done to device **inen()** is handled differently than system dumps to other devices. The **inen** dump handler always automatically reboots the system with a boot path of **inen() −d**. This path instructs the bootstrap to do the system dump across the network to the server. This means that the autoreboot state is ignored when a system dump is taken to device **inen()**. Additionally, since the kernel is always booted with a boot path of **inen() -d** after such a system dump, any boot path that has been set through **dg_sysctl**(2) is ignored, and **inen() -d** becomes the default boot path for the new kernel. The consequences of this are that all future automatic reboots of the system also cause a system dump to be done, and this may overwrite the previous system dump done by the client. If no panic dumps are done by the client, the autoreboot state is honored.

5.1.19 Default Dump Device

If you are using AV 530, AV 4600, or AV 5000/6000 series systems, you should not use the installer kernel's default dump device. The default dump device is **st(insc(),4)**. You should use the **dg_sysctl**(1M) command to modify the default dump path. If your system panics and you try to take a dump with the default dump device, then the system will hang for about 20 minutes. It then comes back up and asks if you would like to try the dump again.

5.1.20 Entering the Wrong Dump Device

When you are taking a system dump, be sure you enter the correct dump device at the "Dump destination device?" prompt. If you enter the wrong dump device at this prompt, the dump fails and your system hangs. The system cannot recover in this case, and you must reset your machine to get the SCM prompt.

5.1.21 System Shutdown Message

When you are shutting down your system on a multi-processor system, the message displayed telling you that SINGLE USER MODE has been reached is occasionally truncated by one line. If this should occur, there is no problem with your system. The system will shutdown safely and normally.

5.1.22 Emergency System Shutdown

You can perform an emergency system shutdown from the system console, which is either a terminal on the **duart** controller or the workstation graphics console. The **syscon** device driver panics the system with panic code

```
030000013 (TS_PANIC_OPERATOR_SHUTDOWN)
```

when it detects the following hot key character sequence: ^]^[^]^[^]^[. The caret (^) indicates you type the next character ("]" or "[") while holding down the <CONTROL> key. You must type this sequence in order with no intervening characters.

For "soft" system hangs, where the console driver is operative, you can use the hot key sequence to initiate a system dump. To initiate a system dump from a "hard" system hang, reset the system using the reset switch, then enter:

```
S 1000
```

at the SCM prompt. Note that you cannot use this to produce a system dump from a system that hangs after the panic sequence has been entered.

In case **inen()** did not work as a dump destination device with a diskless workstation, you can still perform a dump by rebooting the machine with a **-d** flag at the SCM prompt with the following command:

b inen() -d

Note that if you use the hot key character sequence, the system's autoreboot flag is set to "halt" and the autodump flag is set to "ask".

5.2 Programming Commands

5.2.1 Nonstandard Programs

Some nonstandard programs dereference NULL pointers and expect address 0 to contain 0. You can make these programs work in both the ELF and COFF environments.

When linking an ELF executable, add the option **-z lowzeros** to your **cc**(1) or **ld**(1) command line (see the **ld**(1) manual page).

When linking a COFF executable, you must create a file containing the following script and include that file among the arguments to **cc** or **ld**.

```
SECTIONS {
   .text  BIND(0x0) BLOCK (0x10000):
   {
      . += 0x1000;
      *(.init)
      *(.text)
      *(.initp)
      *(.finip)
      *(.tdesc)
   }
   GROUP BIND(
      ((((ADDR(.text)+SIZEOF(.text)-1)/0x400000)+1)*0x400000)
      + ((ADDR(.text)+SIZEOF(.text))%0x10000) ) :
  {
      .data   : { }
      .bss    : { }
}
```

5.2.2 Known Problems

The following are known problems in this release:

- The System V Release 4 archive utility (**ar**) accepts COFF and ELF objects separately, but cannot operate on mixed COFF and ELF objects.
- The disassembler does not produce line information when the debugging switch is provided.
- The profiler does not produce line information when the debugging switch is provided.

5.2.3 cc -X[tac]

Note that the **cc**(1) command invokes the GNU C compiler, not the AT&T CI5 compiler. In particular, you should be aware that the behavior of GNU C under the **-X[tac]** options approximates but does not duplicate the behavior of CI5 under the same options.

5.2.4 dbx and sdb

The following problems apply to both the **dbx**(1) and **sdb**(1) debuggers:

- When stepping to a routine that is an ELF shared-object, **dbx** and **sdb** produce the following warning message:

  ```
  Note: No debugging information at the current position.
  This region will be treated as one line.
  ```

  This occurs even if the routine is built for debugging. If there is no debugging information for the shared-object routine, the debugger single-step executes at the instruction level until the program executes code with debugging information. In this case program execution slows dramatically, so it may take a long time for the debugger to regain control. To avoid this problem, use the "next" button or macro.

- Avoid issuing the **step** command when a shared ELF executable (an executable participating in ELF dynamic linking) has been loaded initially. When you issue this command before the process has been initialized, the debugger single-step executes at the instruction level through all the initialization of **libc** and other statically known shared-object dependencies. This process can be very lengthy.

- The C and Fortran Language Processors use **malloc()** to allocate storage for some values during routine invocation. C also uses it for some string-literals. These operations fail with a segmentation violation if executed before a shared ELF executable has initialized.

- If you set breakpoints in any shared-object, you cannot run the program again in the same debugger session.

- If an invalid address is displayed or the contents of an address are displayed using an invalid format, the following error message may be displayed:

  ```
  Internal Error: assertion "address_segment!=NULL" failed at line. . .
  ```

  This does not affect execution of the debugging session.

The following problems apply only to the **dbx** debugger:

- The **dbx** debugger accepts both ELF and COFF executables. However, if you debug executables of different file formats in one **dbx** session, the debugging environment (for example, aliases and debugger variables) is lost at each change of format. To avoid this, use the **save** and **restore** commands to preserve the environment across the change to a new file format. L1 Since **dbx** is no longer built using **gdb**, the debugger can no longer be shared.

- For dynamically-linked executables, if you use a **set** or **assign** command to assign a new string value to a variable declared as a pointer to a character string, a segmentation fault occurs.

- When debugging an ELF executable, aliasing a command to a unique prefix of itself causes the command functionality to be lost.

The following problems apply only to the **sdb** debugger:

- During debugging of an ELF executable, if you set the verbose mode to level 2 or greater and a large number of instructions are stepped through, your program may core dump.
- During debugging of an ELF executable,if a user-defined procedure is executed using the debugger's procedure call capability, the following warning message be displayed:

  ```
  Internal Error: Expression tree at. . .has no value
  ```

  This does not affect execution of the debugging session.

5.2.5 Corrupt Object Files due to Relocation Entries

If an object file is corrupted such that a relocation entry has a zero relocation type, the **ld**(1) link editor issues these warning messages:

```
ld: relocation entry found for non-relocatable symbol in. . .
ld warning: Output file <filename> not executable
```

In this situation, **ld** returns an exit code of zero, although the output file is not executable.

If this corrupt executable occurs in a module that is being built into the kernel, the kernel build may appear to have succeeded. Check **/var/Build/system.***module***.build** for error messages to ensure your executable is not corrupt.

5.2.6 Layered Products

Some of your layered products may require more time to compile and link in DG/UX Release 5.4 than in release 4.3x. A patch is available to correct this problem.

5.3 The sysadm Command

5.3.1 Auto Configuration Operation

The System -> Kernel -> Auto Configure operation asks whether you want to overwrite the existing system file. If you are upgrading and want to save your previous system file, you must do so manually.

The System -> Kernel -> Auto Configure operation does not configure a dump device into the kernel. If you invoke the key sequences necessary to take a system dump you are prompted with a default dump device. You are also given the option of specifying a device other than the default. You may also set the default dump device with **dg_sysctl**(1).

5.3.2 Adding a Printer

The method for adding a printer via **sysadm** has changed significantly from previous releases of DG/UX. See "Printing Now" in Chapter 11 in *Managing the DG/UX™ System* (093-701088) and Chapter 6 in *Customizing the DG/UX™ System* (093-701101) for examples of how to add a printer in DG/UX Relase 5.4.

5.3.3 Adding TTY Lines

If you want to add more than 255 TTY lines in the **sysadm** Device Port Terminal menu, you must add the lines in groups of 255 or less. If you try to add more than 255 lines in a single group, the operation fails and the system core dumps.

5.3.4 Adding and Modifying Groups

If you have set up different file systems for **/etc** and **/tmp**, then some **sysadm** group operations do not work. The Add Group operation adds the group and its id, but no users. The Delete Group operation returns an error message. The Append function of the Modify Group operation also does not work in this case. This problem will be fixed in the next release of DG/UX.

5.3.5 UUCP Device Management

When the **sysadm** UUCP device management function adds devices to the **/etc/uucp/Devices** file, it leaves the ",M" off of the TTYs connected to modems. You should make this addition manually to the appropriate entries in the Devices file. You cannot dial out through a modem unless the ",M" is added to its entry in the file.

5.3.6 Restoring Pre-5.4 Backup Tapes

When restoring a backup tape that was made using a pre-5.4 release of DG/UX, you must choose the medium type "pre_5.4." The other medium types do not work correctly because they assume 16K byte block size. Prior to release 5.4, **dump2**(1M) used a 10K byte block size.

5.3.7 OSF/Motif Interface

To use **sysadm**'s OSF/Motif interface, the DG/UX X Windows System package must be loaded and set up. Note that you must have the X Windows package mounted for **sysadm** to work.

5.3.8 xsysadm Warning Messages

When an **xsysadm** warning message window is displayed, you might experience problems using the "close" button to close the window. In this case, press the Enter or New Line key to remove the warning message and continue normal processing.

Xsysadm is not hung when this problem occurs. The program is waiting for a response to the "Hit <enter> to continue" prompt in the *Interactions* window. When a warning message is displayed, the *Interactions* window and its prompt are hidden by the main **xsysadm** window.

5.4 Other Administrative Commands

5.4.1 diskman

The stand-alone **diskman**(1M) program was originally designed to load DG/UX only. In release 5.4, **diskman** enables you to load X Windows at the same time as DG/UX. However, **diskman** does not support software package installation after you have loaded the DG/UX package. If you try to load other packages with **diskman** after DG/UX is loaded, the installation may seem to work correctly, but it will be incomplete. Once you have loaded the DG/UX package, all software installation must be done through **sysadm**.

5.4.2 halt

Under the following conditions, issuing the **halt -q** command may hang your system:

- You have the model 30 8 line cluster box with no printer attached to the parallel port, and

- You have enabled **ttymon** and told it to monitor "all" lines

To ensure there is no loss of data, you should follow the normal procedures of running shutdown prior to the **halt -q**. You should also wait a time no less than **MAXBUFAGE** (default 60 seconds) to allow the system to flush all the buffers. In this case, the hang on **halt -q** does NOT result in data loss.

5.4.3 lpr and lpd

Data General recommends that you do not use the BSD line printer spooler (**lpr**(1) and **lpd**(1M)) due to recently discovered security problems.

If you want to allow for printing from other BSD-based machines, such as Sun machines, you should use the BSD emulation mode now supplied by the new 5.4 AT&T **lpsched**(1M) scheduler. This scheduler is not vulnerable to these security problems.

Data General is working closely with Berkeley on supplying a BSD line printer spooler that corrects these security problems. We will make it available as a patch as soon as possible for customers who require the actual BSD line printer spooler.

5.4.4 lsd

You cannot load DG/UX 5.4 kernel dumps with the 4.3x version of **lsd**(1). The dump file format has changed in 5.4 and the 4.3x **lsd** does not recognize the new format. However, you can load 4.3x dumps with the 5.4 **lsd**.

5.4.5 osysadm

The **osysadm**(1M) command will not be supported in future releases of the DG/UX system.

5.4.6 ps

The **ps**(1) command does not display the nice value for processes when the **-l** and **-f** options are used at the same time. The "NI" column heading is displayed, but the nice values are not. The **ps -l** command reports the nice values, **ps -lf** does not. This will be fixed in the next release of DG/UX.

5.4.7 reboot

If an operating system client machine has panicked and rebooted and you later execute the **reboot**(1M) command with no arguments, then another panic dump may be written. This new dump could overwrite the previous panic dump. To avoid this problem, you can use the **dg_sysctl**(1M) command to set the default boot path, or you can invoke **reboot** with an explicit boot path (such as **reboot inen()**).

5.5 General Commands

5.5.1 bind

You cannot **halt** your system until any **bind**(2) operations that are in progress have completed. After you shut down your system, be sure to check for any still running **bind** processes. Do not attempt a **halt** until all **bind** processes are safely completed.

5.5.2 cpio

The **cpio**(1) command no longer supports the **-h** switch. The **-b** and **-S** options of **cpio** may cause core dumps. You can use these byte swapping options, but if you get a core dump try issuing the command without them.

Also, when **cpio -o** encounters a control point directory (CPD), it writes the directory limit size into the "file size" field of the **cpio** file header. This conflicts with common usage and the IEEE POSIX standard (1003.1-1988 and 1003.1-1990), which require this value to be zero for directories. If a CPD is dumped to the archive, any archives made with DG/UX 5.4 (and 4.32) **cpio** can be read by neither DG/UX 4.31 (or earlier) **cpio**, nor by other vendors' **cpio** implementations. To avoid this incompatibility, do any of the following:

- Move the contents of each CPD you want to archive into a regular directory, and archive that directory instead

- Use some other tool for creating the archive, such as **tar**(1)

- If a CPD absolutely must be dumped in **cpio** format, use the **cpio** executable from DG/UX

4.31

5.5.3 cscope

If you use the **-i** option to the **cscope**(1) command, only indicate the name of the file containing the list of source files you want to browse. If you enter any individual source file names on the command line, the **-i** option is ignored. This will be fixed in the next release of DG/UX.

5.5.4 editread

The **editread**(5) command line editor may exhibit degraded performance relative to previous DG/UX releases. Ths is especially true on systems with tight memory resources or heavy file paging activity. Also, the changeover to STREAMS-based TTY for System V release 4 compliance has made **editread** type-ahead less effective and may result in occasional lost characters when you use type-ahead.

The **editread** command is not supported in the **ksh**(1).

5.5.5 fmt

The **fmt**(1) command truncates input lines longer than 1024 characters without warning. This will be fixed in the next release of DG/UX.

5.5.6 idi

When you use the **idi**(1) OSF/Motif interface, each operation is limited to 8 screens, each screen is limited to 8 querygroups, and each querygroup is limited to 16 queries.

5.5.7 ln

You can no longer overwrite an existing link with the **ln**(1) command. For example, suppose you have an existing link "doc -> /usr/doc" that you want to replace with "doc -> /etc/doc." Prior to DG/UX Release 5.4, you could issue the command **ln -s /etc/doc doc** to overwrite the existing link with the new one. If you issue this same command in release 5.4, you receive the following error message:

```
ln: cannot create doc
ln: file exists
```

The command will not overwrite the existing link. You need to remove the existing link and replace it with the new one.

5.5.8 mail

The format of the DG/UX Release 5.4 mailbox has changed. In release 5.4, **/bin/mail** puts a new field, "Content-Length", into the message header. This field indicates the number of bytes in the mail message. The **mail**(1) command uses this field to separate mail messages when reading mail. Other mail reading programs that are still using the "From" line as a new message indicator are no longer compatible with the release 5.4 **mail**. You should modify these programs accordingly.

5.5.9 pwd

When you mount a blank 5.25-inch DOS-formatted floppy disk as a DOS file system, there is a problem using **pwd**(1). When you use **mkdir** to make a directory on the disk and use **cd** to move into the new directory, **pwd** returns the following error message:

```
pwd: cannot determine current directory!
```

The directory does exist and **cd ..** still works.

5.5.10 sh

The **sh**(1) command programming language no longer contains the **limit** command. **Limit** is not a command in either the DG/UX Release 5.4 or the AT&T System V Release 4 Bourne shell. The **limit** command will be added to the 5.4.1 release of DG/UX. For more information, see the **sh**(1) man page.

5.5.11 stty

The DG/UX Release 5.4 **stty**(1) and **berk_stty** commands now support all BSD command line syntax, including **stty everything**, **stty all**, and **stty crt**. However, they have not been changed to produce BSD style output, because the limited BSD syntax would hide many features of the new unified STREAMS-based line discipline.

You should also note that **stty** no longer accepts the character **u** to undefine special characters. If you are using the **stty** *special character* **u** command in your **.login** or **.cshrc** scripts to undefine special characters, you need to change those scripts.

5.5.12 tar

The **tar**(1) command no longer supports the **-B**, **-C**, and **-O** switches.

5.6 Time Zones

You use the environment variable **TZ** to define the local time zone. You normally manipulate this variable via the **sysadm** menu selection System -> Date -> Set. Prior to this release of the DG/UX system, **TZ** was assigned values that looked like **EST5EDT** (Eastern Standard Time, 5 hours West of Greenwich, Eastern Daylight Time during the summer). The commands and libraries included in this release of the DG/UX system understand both this traditional style of **TZ** value, and also a new, 'zic' style. The zic-style values look like **:US/Eastern**. The advantage of this style is it is capable of supporting a far more complete set of time zone rules, covering the whole world. These rules include subtleties such as switching in and out of Daylight Savings Time on exactly the right day in each year.

Application programs that are linked on version 5.4 of the DG/UX system will run correctly. On DG/UX Release 5.4 installations that use the traditional style **TZ** values, all applications, no matter when or where they were linked, should work correctly with regard to time zones.

However, there is a potential problem with the combination of the new style of **TZ** values and application programs (including third-party software) that were not linked on DG/UX 5.4 systems. Applications that were linked on non-DG/UX machines may work perfectly, or may exhibit any of the following undesirable behaviors.

Applications that were linked on DG/UX systems prior to release 4.30 include code that will not recognize the zic-style **TZ** values at all. When this code encounters such a value, it falls back to displaying times in GMT.

Applications that were linked on 4.3x releases of the DG/UX system include timezone-handling code that might not function correctly when it encounters a zic-style **TZ** value. The following problems might occur:

- Running slowly
- Consuming progressively more and more memory space

If your application code does not manipulate dates and time much, then you might not detect any observable problems. However, if your application uses the date/time manipulation subroutines heavily, you may observe poor performance, or the application may grow until the system reports a lack of swap space.

Because of these potential problems, this release of the DG/UX system is shipped with a default **TZ** value of **EST5EDT**, and North American users should continue to use this traditional style. Users in other parts of the world can use variations on the traditional style that are described in the **environ**(5) man page, or they may use the zic style.

If you use a zic style **TZ** value as the default for your system, and encounter one of the problems described here, you can change your default value (using the **sysadm** command). Alternatively, the value of **TZ** can be changed for an individual user by editing the user's **.profile** file (for Bourne shell users) or **shell users). See the files /etc/TIMEZONE** and **/etc/TIMEZONE.csh** for examples of the correct syntax. You can also use this syntax in other shell scripts to set the value of **TZ**.

5.7 Libraries

5.7.1 curses Library

The **curses** library has been brought into compliance with the 88000 Object Compatibility Standard (OCS).

5.7.2 editread Library (libeditread.a)

Programs linking in the **libeditread.a** library (**-leditread**) must now link in **libgen.a** (**-lgen**) and **libw.a** (**-lw**), in addition to **libcurses.a** (**-lcurses**). These changes are required for two reasons. First, **editread**(5) uses regular expression functions, which have moved to **libgen.a**. Second, **editread** has been enhanced to support international code sets, which require wide character support from **libw.a**.

5.7.3 Standard I/O (stdio)

The stdio routines in **libc** now automatically set their buffer size to an optimal value for *sequential* I/O. If your application is doing a lot of *random* I/O and reading small amounts of data, you should use the **setvbuf**() function (see the **setbuf**(3S) man page). Using **setvbuf**() to buffer data at a size closer to the average amount read/written will probably increase performance.

If you enable the imprecise floating point exception (**FP_X_IMP**), you might encounter some problems when using the stdio functions (such as printf/scanf) in **libc**. For example, these problems might occur when you are working with very large or very small numbers at the boundaries of limits. In both the ELF and COFF environments, the imprecise floating point exception is disabled by default.

POSIX 1003.1 specifies that **fflush** and **fclose** on STDIO read STREAMS should adjust the file pointer of the underlying file description to match the position within the buffer. Due to this switch to POSIX behavior for **fflush** in the **m88kdguxelf** environment, some programs may experience problems. The problem can occur where a process using STDIO does a fork, and the child process does an **fclose** on a read STREAM that the parent is to use later. A subsequent read (such as **fread**, **getc**, etc.) done by the parent is likely to reread a portion of the current buffer. This happens when the parent reaches the end of the buffer that was in use when the fork was performed. To avoid this problem, you should call **fflush** just prior to the fork (on any STDIO read STREAM on which the child is executing **fclose**).

5.7.4 Values Files

To implement correct default behavior for math functions in the absence of any **-X** switch to the **cc** command, each version of **libc** in the various environments should have one "values" file (values-Xa.o, values-Xc.o, or values-Xt.o) in it. Two versions of **libc** have the wrong default values.

All three "values" files are contained in the static **libc.a** in the **m88kdguxelf** environment. This results in the default behavior for math routines being ANSI rather than 'traditional' (values-Xa.o appears first in the library). The desired default (that is, the one that matches the shared version of **libc**) is 'traditional'. If you are building static ELF executables and using **ld** directly (this is not a problem when using the **cc** or **gcc** command to invoke **ld**), then you should specify the "values" which you desire explicitly in your **ld** command line (for example, **ld -dn -o foo /lib/crt0.o /lib/values-Xt.o foo.o -lc**).

The wrong default "values" file is provided for the **m88kbcs** and **m88kocs** environments. The default "values" file is **values-Xt.o** for 'traditional'. If you are using the **m88kbcs** or **m88kocs**, you should use the **values-Xa.o** for partial ANSI. You should use the **-Xa** switch with **cc** or **gcc**. If you are using **ld** directly then you need to include **values-Xa.o** explicitly in your **ld** command line.

5.7.5 libm

Due to a problem in **libm**, you should not start using the new functions that take float arguments. The problem will be corrected in a future release of DG/UX. The following functions are affected:

acosf **asinf** **atan2f**
atanf **ceilf** **cosf**
coshf **expf** **fabsf**
floorf **fmodf** **log10f**
logf **powf** **sinf**
sinhf **sqrtf** **tanf**
tanhf

5.7.6 terminfo Directory

The default **terminfo** directory has changed from **/usr/lib/terminfo** to **/usr/share/lib/terminfo**.

5.8 Printer Services

5.8.1 LP Subsystem

You should be aware of the following concerns when using the LP printer subsystem:

- For printing from a DG/UX Release 5.4 system to a pre-5.4 print server, a user must be able to **remsh** *remote system command* successfully. See section 4.7 for details.
- The new 5.4 to 5.4 **lpNet** networking feature is available, although an occasional suspension of LP Network activity may result. In the event of a suspension, the administrator must restart **lpsched**(1M), possibly on both the local and remote systems. Note that you may still use the pre-5.4 method of remote printing, even between 5.4 equipped schedulers.

- In order for the new networking feature of Release 5.4 to work, the correct services entry must be in place for the Network Listener.

  This services entry is contained in **/etc/services**, if you do not use the Network Information Services (NIS). If you do run NIS, you need to have the Network Administrator place the correct entry into the **services.byname** NIS map. The correct entry is shipped in **/etc/services.proto**, and is:

```
listen     2766/tcp     serve # network listener port
```

- The **lpNet** networking feature might fault when attempting to network with a BSD system. If you need to do a lot of printing to remote BSD systems, the administrator may choose to use the actual BSD Line Printer Spooler (**lpd**) for those printers, instead of the new BSD emulation modes of the 5.4 scheduler. In this case, be certain to read the area of the release notice concerning the security problems with the **lpd** command.

  Note that by default the AT&T **lpsched** BSD emulation is listening to the BSD printer network port. If you desire to use the BSD **lpd**, you must delete the Port Services entry for this emulation mode. This can be accomplished through **sysadm** with the sequence Device -> Port -> Port Service -> Delete -> tcp -> lpd.

5.9 Device Services

5.9.1 Line Settings of TTY Modes

The default line settings of TTY modes have been changed to conform to the SVID standard. This can cause problems with applications using the old defaults. You can fix any problems that occur by adding code to the application program to explicitly set each TTY mode.

5.9.2 Terminal Lines

You should not run **ttymon** on terminal lines that are connected to other computers. Normally terminals lines have active and passive ends. The computer is active and the terminal is passive. If you connect two active lines together, your system might panic.

5.9.3 Designating Terminals as Administrative Devices

You can designate a subset of terminals as administration devices, which are available in run level 1 while other terminals remain disabled. To get additional terminals to run at **init**(1M) state 1, you must add a **ttymon**(1M) entry to the **/etc/inittab** file for each additional terminal. These entries should be similar to the "con" inittab entry. See Chapter 3 in *Managing the DG/UX™ System* (093-701088) for more information on the **/etc/inittab** file.

5.9.4 Reloading the Syac for a Set of TTY Lines

If you reload the **syac** for a set of TTY lines that are being monitored by an active **ttymon** port monitor, **ttymon** might deactivate the lines. To reactivate the lines, enter the following command:

```
sacadm -x -p port-monitor-name
```

This causes the port monitor to reread its database file and open any deactivated TTY lines.

5.9.5 Systems with a High TTY Count

On systems with a high TTY count, you may notice that one or more TTY devices are not enabled when the system is booted to **init** state 2. This problem occurs if **init** starts the **ttymon** process for the ports before the loading of the **syac** has completed. To enable any failed TTYs, issue the following command:

```
sacadm -x -p port-monitor-name
```

One definite indication this problem has occurred is when the log file for the **ttymon** (**/var/saf/***port-monitor-name***/log**) contains messages that state the service for the TTY is respawning too rapidly.

5.9.6 The VCD/8P Cluster Controller and ttymon

The VCD/8P cluster controller provides eight RS-232C TTY ports and one parallel printer port. The parallel printer port is accessed in the same way as the TTY ports through a device node of the form "/dev/ttyNN". However, even though the formats of the names are the same, you must not run a **ttymon**(1M) port service on the printer ports. Otherwise the system may hang when you try to **halt** it. See the section on the **halt** command in this release notice for details.

In particular, when using **sysadm**(1M) to add terminals, you should not specify "all" as the set of terminal lines to configure. Instead, use the procedure described in Appendix C of *Customizing the DG/UX™ System* (093-701101) to determine which "/dev/ttyNN" devices on your system are really TTY ports. Then specify only the true TTY ports to **sysadm**, using the range list format described in the section "Adding a Group of Identical Terminals" in the same manual.

5.9.7 Modems and the ttymon Wait-Read Count

Modems that send status messages to the host computer, such as "RING" and "CONNECT", may "chatter" when attached to a DG/UX Release 5.4 system. The modem's transmit and receive indicators flash almost continuously, and the modem may fail to answer calls. These symptoms indicate that the **ttymon**(1M) wait-read count is not set correctly for the port service of the modem.

The wait-read count is the number of newline characters that must be received on a port before **ttymon** presents a login prompt on that port. The **sysadm**(1M) and **ttyadm**(1M) default for the wait-read count is 0, which indicates that the prompt will be sent immediately. This default is proper for terminals and "quiet" modems. However, it does not work for modems that send status messages. The first **ttymon** login prompt is interpreted as a (bad) command by the modem, which responds with a status message, which causes another prompt from **ttymon**, and so on.

For modems that send one "RING" message, one "CONNECT" message, and two blank lines (all terminated with a carriage return and line feed), you should set the wait-read count to 8. This recommendation covers Hayes®-compatible modems set to transmit status messages.

5.9.8 Modem Signals and ttymon

Some configurations of modems and terminals that worked in previous DG/UX releases with **getty**(1M) or **uugetty**, may not work in DG/UX Release 5.4 with **ttymon**(1M). The modem or terminal never receives the Data Terminal Ready (DTR) signal from the system, and therefore never goes on-line or answers calls. This problem results from modem signal "handshaking" added to **ttymon**.

To guarantee reliable hangup and initialization of modems, **ttymon** requires EIA RS-232C control signal protocol from Data Communications Equipment (DCE) devices, such as modems, attached to ports monitored by **ttymon**. This requirement also applies to Data Terminal Equipment (DTE) devices emulating DCE devices, for example, terminals cabled so that they respect RS-232 control signals from the host instead of jumpering the signals together at each end of the cable. Specifically, **ttymon** requires that the DCE device respond to the loss of the DTR signal by lowering the Data Carrier Detect (DCD) and Data Set Ready (DSR) signals.

If you encounter problems with **ttymon** and modem signals, you should reconfigure your devices and/or cables so they implement the control signals specified by the EIA RS-232C standard. There are often hardware switches or software commands on modems and terminals to make the device respect DTR by lowering DSR and DCD. For example, on a Hayes SMART-MODEM 1200™ or compatible, set DIP switch 6 up; on a Hayes SMART-MODEM 2400™ or compatible, issue the command "AT&C1&S1&W".

If you cannot change your hardware settings, then you can alternately disable the "hangup" normally done by **ttymon** for the port service. To implement this, start **sysadm**, enter the dialogue "Device -> Port -> Port Service -> Modify", select the appropriate port service, and then answer "no" at the "Hangup?" prompt (this corresponds to the **-h** option of **ttyadm**(1M)). Another alternative is to use a separate **ttymon** process in express mode to monitor each affected port. See the **ttymon**(1M) and **getty**(1M) man pages, and section 5.9.3 in this release notice, "Designating Terminals as Administrative Devices", for more information.

6 Documentation

This section contains information on the titles and changes in the DG/UX Release 5.4 documentation set. It also contains a list of the DG/UX and other UNIX training classes available from Data General Educational Services.

NOTE: When you are ordering new manuals for DG/UX Release 5.4, be sure to include the revision number with your order. The revision number is the last two digits in the manual's part number.

6.1 Titles

The *Guide to AViiON® and DG/UX™ System Documentation* provides a complete list of DG/UX and all other AViiON documentation. An on-line version of this guide is in the file **/usr/release/doc_guide**. The on-line version does not include illustrations.

6.2 Changes

The following manuals are new or revised for Release 5.4:

Data General Manuals

Customizing the DG/UX™ System (093-701101-00)
Guide to AViiON® and DG/UX™ System Documentation (069-701085-00)
Installing the DG/UX™ System (093-701087-00)
Installation Roadmap (069-701084-00)
Managing ONC/NFS® and Its Facilities on the DG/UX™ System (093-701049-03)
Managing TCP/IP on the DG/UX™ System (093-701051-04)
Managing the DG/UX™ System (093-701088-00)
Porting and Developing Applications on the DG/UX™ System (069-701059-03)
Programmer's Reference for DG/UX™ System (Volume 1) (093-701055-02)
Programmer's Reference for DG/UX™ System (Volume 2) (093-701056-02)
Programmer's Reference for DG/UX™ System (Volume 3) (093-701102-00)
Programming in the DG/UX™ Kernel Environment (093-701083-00)
Programming with TCP/IP on the DG/UX™ System (093-701024-02)
System Manager's Reference for the DG/UX™ System (093-701050-02)
User's Reference for the DG/UX™ System (093-701054-02)

Vendor Documents

Character User Interface (FMLI and ETI) (093-701107)
The KornShell Command and Programming Language (093-701105)
Programmer's Guide: ANSI C and Programming Support Tools (093-701104)
Programmer's Guide: STREAMS (093-701106 Hall)
Programmer's Guide: Systems Services and Application Packaging Tools User's Guide (093-701103)
X/Open Portability Guide (Purchase from Prentice Hall)

Obsolete Manuals

The following manuals are obsolete for Release 5.4:

Installing and Managing the DG/UX™ System (093-701052)
OSF/Motif Application Environment Specification (069-100326)
OSF/Motif Programmer's Guide (069-100323)
OSF/Motif Programmer's Reference (069-100325)
Programming in the DG/UX™ System's Application Environment (093-701076)
STREAMS Primer for the DG/UX™ System (069-701033)
STREAMS Programmer's Guide for DG/UX™ (069-701034)
Using the DG/UX™ Software Development Tools (093-701078)

6.3 Training Classes

DG/UX Release 5.4 and other UNIX training is available from Data General Educational Services. If you are within the United States or Canada, call 1-800-343-8842 for further information. If you are outside the United States or Canada, ask your Data General Representative for the appropriate phone number. The following classes are available:

S205 *UNIX User* (5 days)
S288 *DG/UX System Administration for AViiON Systems* (5 days)
S125 *C Programming* (5 days)
S181 *Advanced UNIX Script Writing*
S182 *Tailoring Looking Glass & OSF/Motif Environments* (2 days)
S188 *DG/UX Operations for AViiON Systems* (5 days)
S315 *DG/UX System Programming* (5 days)

SM101 *UNIX:Security Issues* (1 day)
SM133 *C++ Programming Workshop* (5 days)
SM482 *Writing DG/UX Device Drivers* (4 days)
SM488 *X Window Programming for C Programmers* (3 days)
SM588 *OSF/Motif Programming for C Programmers* (3 days)
SM490 *Network Interoperability* (4 days)
SM510 *DG/UX Tuning Workshop* (3 days)
SM580 *A Survey of Graphics Programming Techniques* (3 days)

Using UNIX: Computer-Based Training Course
Using the vi Editor: Computer-Based Training Course
Writing Bourne Shell Scripts: Computer-Based Training Course
DG/UX System Administration: Computer-Based Training Course
Loading and Generating DG/UX: Computer-Based Training Course
Using TCP/IP: Computer-Based Training Course
UNIX Network Management: Computer-Based Training Course

7 Software Distribution

7.1 Media

The DG/UX System is included in the following packages:

- Model Number P001, the **DG/UX Operating System with X Windows** package (079-600183-00)

- Model Number Q001, the **DG/UX Operating System** package (079-600182-00)

7.2 Files

A list of files that are loaded when the DG/UX system is loaded on your disk is in the file **/usr/release/dgux_5.4.fl**. A list of files loaded when GNU C is loaded is in the file **/usr/release/gcc_1.37.31.fl**.

8 Installation Instructions

The installation process for DG/UX Release 5.4 is significantly different from previous releases. See *Installing the DG/UX™ System* (093-701087) **before** you begin installing release 5.4. This is a new manual that guides you through installing DG/UX Release 5.4.

The previous installation manual, *Installing and Managing the DG/UX™ System*, is now obsolete. The information in this manual has been divided among three manuals: *Installing the DG/UX™ System*, *Customizing the DG/UX™ System* (093-701101), and *Managing the DG/UX™ System* (093-701088). See *Guide to AViiON® and DG/UX™ System Documentation* (069-701085) for more information on these and other documentation changes.

8.1 Notes

You should be aware of the following things before you install DG/UX Release 5.4:

- DG/UX Release 5.4 does not fit on a 179 megabyte disk. You need at least 322 megabytes.

- *Customizing the DG/UX™ System* (093-701101) has an error on page 8-27. This page contains a procedure for "Loading the Root Logical Disk on the Local Physical Disk." In step one of that procedure, the manual tells you to boot to init level S with the command **b inen(0) -S**. The command should actually be **b inen(0) -1**. If you boot only to init level S, the **tar** command in step five fails.

- You should load DG/UX Release 5.4 only on either a clean disk or a completely installed DG/UX Release 4.3x disk. These are the only two loading configurations supported in Release 5.4. If you try to install Release 5.4 on a disk that contains an incompletely installed copy of Release 4.3x (that is, some packages were loaded but not set up), you might have problems with leftover 4.3x setup scripts. Use the 4.3x **sysadm setuppackage** command before installing 5.4 to see if any packages have not been set up.

- Do not substitute a DG/UX Release 4.3x **/etc/inittab** file for a DG/UX Release 5.4 **/etc/inittab** file. If you do this, your system will be severely crippled. For example, the STREAMS TTY drivers are not loaded and a workstation's console becomes unusable.

- The default disk size recommendations in the installation program assume that you have not put additional files into your **/** and **/usr** file systems. If **/** or **/usr** contain files that were not delivered by Data General, then the program has no way to determine how large your file system should be. To determine the approximate size you will need, add the number of blocks you are using beyond the default DG/UX Release 4.3x configuration to the default size of the DG/UX Release 5.4 configuration. For example, if you are using 40,000 more blocks than the default 4.3x configuration, you will probably need 40,000 more blocks than the default 5.4 configuration to complete the installation.

- Beginning with release 5.4, DG/UX is shipped on release tapes in compressed format. If you receive the error message "stdin: not in compressed format" during installation, the system has detected an error while reading the tape. This usually means that your tape heads are dirty. If you receive this error, you should clean your tape heads and read the tape again.

- If you are loading this release into a secondary release for operating system clients when the operating system server is not running DG/UX Release 5.4, see Chapter 9 in *Customizing the DG/UX™ System* for special instructions. Your server must be running DG/UX Release 4.32 or later to have an operating system client running release 5.4.

- If you are upgrading from a DG/UX release prior to 4.32 to DG/UX Release 5.4, you should refer to the *Release Notice: DG/UX™ X Window System for AViiON® Systems* before you begin. X11R3 has been upgraded to X11R4, and the installation process is very different.

- When installing DG/UX Release 5.4 on AViiON 5240/6240/7000/8000 series systems or any AViiON system with a HADA or CSS Disk Array subsystem, you need to make the following modification in the **/etc/dgux.params** to ensure event logging.

  — In single user mode, search for the following string in **/etc/dgux.params**:

  ```
  dgsvcd_START="false"
  ```

  — Modify the string to read:

  ```
  dgsvcd_START="true"
  ```

  This ensures the logging of machine-initiated messages to **syslog** via **dgsvcd**(1M). Also, you can optionally forward these messages to designated Data General Support Centers for fast warranty and service contract support.

- If you are installing DG/UX using the stand-alone **diskman** onto a system with the Model 7902 High Availability Disk Array (HADA) subsystem using the Model 6677 Multi-Capacity Tape Drive located in the HADA, you might receive the following error message from the **gridman** utility:

```
there is no currently installed code on this controller
```

In this case, you need to download microcode from a DG/UX or MIDA tape before you can continue using the HADA device. **Gridman** prompts you for the name of the tape drive from which to load the microcode. If you are using the Model 6677 tape drive, you might see the following error message after the microcode has been loaded:

```
System Error in function tape rewind

Error Code = 6 (hex)

No such device or address
```

This error results from the fact that restarting the HADA controller causes a SCSI-bus reset. The reset then starts an open-ended rewind operation on the Model 6677 drive. If you receive this message, wait until the tape drive finishes rewinding and the busy light goes off. Then retry the installation of the microcode. If you try to install the microcode before the tape drive finishes, the open operation fails and any further attempts to install the microcode also fail until you reboot **diskman**.

If you need to reload HADA microcode and the Model 6677 Tape Drive is attached to a **cisc()** adapter, you should use the stand-among version of **diskman** invoked by **sysadm** to load the microcode. If you must use the stand-alone **diskman** and no other devices on the **cisc()** adapter have been used since the last reboot, the first attempt to use the tape drive may fail with an "undefined error" status. To avoid this problem, you should configure the **cisc()** adapter before trying to install the HADA microcode. This can be done, for example, by attempting to register a disk on the **cisc()** adapter that supports the Model 6677 Tape Drive. Whether the disk physically exists or not, this operation configures the **cisc()** adapter. As indicated previously, configuring the adapter forces a SCSI-bus reset, so make sure that the tape drive busy light has gone off before attempting to complete the installation of the microcode.

- To successfully set up DG/UX Release 5.4 after loading into a secondary release area on a 5.4 system, you must manually delete the prep scripts loaded before adding the first client. You need to delete the following files on the server:

  In **/srv/release/***release_name***/usr/root.proto/sbin/setup.d/root**:

  — dgux: **prep_1.r.do**

  — nfs: **prep_1.r.do**

  — onc: **prep_1.r.do**

  — tcpip: **prep_1.r.do**

  In **/srv/release/***release_name***/usr/sbin/setup.d/usr:**

  — X11.doc: **prep_1.u.do**

  — X11.lg: **prep_1.u.do**

- X11.man: **prep__1.u.do**
- X11.sde: **prep__1.u.do**
- X11: **prep__1.u.do**
- aview.man: **prep__1.u.do**
- aview: **prep__1.u.do**
- dgux: **prep__1.u.do**
- gcc: **prep__1.u.do**
- nfs: **prep__1.u.do**
- onc: **prep__1.u.do**

- If you are upgrading a DG/UX 4.3x system that has the NetWare®, OSI/P, NetBEUI, Netbios SDK, or SDLC for AViiON Systems packages installed, you must remove the following files before you shut down and boot from the 5.4 release tape:

  - **/etc/llc.params.proto**
  - **/etc/llc.params**
  - **/etc/llc.rclinktab.proto**
  - **/etc/llc.rclinktab**
  - **/usr/etc/master.d/llc**
  - **/usr/bin/setup_llc_params**
  - **/usr/bin/llc_update_system_file**
  - **/usr/src/uts/aviion/cf/system.llc.proto**
  - **/usr/src/uts/aviion/cf/Libs.llc**
  - **/usr/src/uts/aviion/cf/system.sdcp.proto**
  - **/usr/src/uts/aviion/cf/Libs.sdcp**

  Failure to remove these files will cause problems when it comes time to build a new custom kernel for your system.

- If you are overloading DG/UX Release 5.4 onto a system that has the Lan Manager package installed, you must remove the contents of the following directories before you load 5.4. If you do not remove these directories, you receive error messages during installation. Use the **rm -r** command to recursively remove the directories and their contents.

- **/usr/spool/lp/bin**
- **/usr/spool/lp/logs**
- **/usr/spool/lp/admins**
- **/usr/spool/lp/model**

8.2 Loading Time

The total loading time for all of the packages on the DG/UX Operating System with X Windows tape is approximately 48 minutes. The approximate loading time for each of the packages on the tape is given in the following table:

dgux	26 minutes	dgux.man	4 minutes
gcc	1 minute	gcc.man	<1 minute
tcpip	1 minute	tcpip.man	<1 minute
nfs	1 minute	nfs.man	<1 minute
onc	1 minute	onc.man	<1 minute
X11	7 minutes	X11.man	2 minutes
aview	1 minute	aview.man	<1 minute
X11.sde	1 minute	X11.doc	1 minute
X11.lg	<1 minute		

The time to load these packages may vary because of different combinations of equipment, system activity, and the method used to load and install the packages from the tape. These times were derived from an installation on an AViiON AV4000 workstation with a 600MB SCSI disk, using **diskman**(1) to load the packages from the tape.

8.3 Log Files

The DG/UX 5.4 **sysadm** utility keeps a record of the events that occur during the installation and setup process and stores that information in log files. The utility keeps the following log files:

/var/adm/log/installman.log

A record of all actions initiated by **installman**(1M) during the installation process. These actions include package setup, kernel building, and automatic rebooting.

/var/setup.d/log/*pkg***.root**

A record of actions performed by the root (host-specific) portion of the setup for a package *pkg*. For example, the log file for the **tcpip** package is **tcpip.root**.

/var/setup.d/log/*pkg***.usr**

A record of actions performed by the usr (host-independent) portion of the setup for a package *pkg*.

/etc/log/init.log

A record of actions performed as a result of run-level changes. This file is overwritten during each boot of the system.

9 Preparing a Software Trouble Report (STR)

If you believe you have found an error in the DG/UX system or its documentation, or if you have a suggestion for enhancing or improving the product, use a Data General Software Trouble Report (STR) to communicate this information to DG.

The standard STR form is available once DG/UX is loaded. It is called **/usr/release/STR_form** and is ready to be printed on a line printer. STR forms are also available from your nearest DG office or DG representative, or the Software Support Center. If your contract permits, you may report the information called for in this section to your Data General representative.

9.1 Gathering STR Information

To help us expedite STR processing, include only one problem or suggestion on each STR form. Please follow these guidelines when filling out your Software Trouble Report:

1. List the name of the product as "DG/UX", model number P001. List the release number as 5.4. If you are running an update or patch, then include its number as well.
2. Indicate what type of STR you are writing. The available types are:
   - Enhancement: describe the proposed enhancement clearly and tell why you want it. The better we understand what you want, the easier it is for us to evaluate your request.
   - Documentation Error: list the title and part number of the document and list the page and paragraph (or section) containing the error. Please state exactly why you think there is an error.
   - Software Problem: clearly and specifically state the problem so that support personnel can try to reproduce it. See the section *Software Problems* below for more details.
3. Provide all of the following information on the STR form:
   - Date
   - Name and release number of the product
   - Release of the operating system
   - CPU type
   - Hardware configuration (if relevant)
   - Names and release numbers of other software running on the system
   - The command line or scenario that caused the problem
   - The action(s) necessary to reproduce the problem

- How often the problem occurs and how serious it is

4. If the problem occurred soon after installing a new release of hardware or software, please note this.

5. If you received an error message, please include the exact text (and number, if present) of the message.

9.2 Software Problems

Report any particular activity or program running on the system that seems to cause the problem. If the program is supplied by DG, report in detail the exact steps used to reproduce the problem. If the program is supplied by another vendor include a copy of the program and its source code if possible. **Report in detail the exact steps used to reproduce the problem.**

9.2.1 System Panics

If your system panics, be sure to **record the panic number**. Then take a dump of the system memory as described below in *Taking System Dumps.*

9.2.2 System Halts or Hangs

If the system hangs:

1. Try the ''hot key'' sequence (see section 5.1.21).

2. If the ''hot key'' sequence does not shut the system down, reset your machine using the reset key or switch. Once in the SCM, enter **"reset"** followed by **"start 1000"**. This invokes the dump sequence. See *Taking System Dumps.*

9.2.3 Incorrect Behavior From a System Call or Device Driver

Write the **shortest possible** program that demonstrates the problem. This can be a shortened version of your actual program, without unnecessary variables and logic. Supply both source and executable copies of this program and a copy of the system image (**/dgux**) with a detailed description of how to reproduce the problem.

9.2.4 Incorrect Behavior of a Command

Write the **shortest possible** shell script that demonstrates the problem. Supply a copy of this script with a detailed description of how to reproduce the problem. Before submitting the problem, be sure that it can be duplicated under ''sterile'' conditions; i.e., with a standard path, standard permissions on files, etc. In other words, try to ensure that your environment is not the cause of the problem.

9.2.5 Fast-Recovery File System

If you need to file a Software Trouble Report (STR) about the fast-recovery file system, please include **/etc/log/fast_fsck.log** in the report. Please also include in the STR all output produced by the fast-recovery **fsck** and the normal **fsck** run that reported problems. When **fsck** is run as part of an **init** script, that output is placed in **/etc/log/fsck.log**. If you're running **fsck** by hand, you should redirect output to a file or pipe it through **tee**.

9.3 Taking System Dumps

If your system panics, you have the opportunity to take a dump of system memory. **In all cases**, when you provide a dump, you should also **provide a copy of the tailored system image** (usually named **/dgux**) that was running at the time of the crash. This image contains vital information necessary for interpretation of the memory dump; the memory dump is useless without the system image. See *Notes On Tapes* below for details on tape format.

Upon system panic, the default behavior of DG/UX is to automatically enter the system dump dialog (that is, autodump is set to "ask"). You may have overridden this with the **dg_sysctl** command. The system may automatically take a dump without prompting the operator (that is, autodump is set to "auto") and the following prompts are not displayed. Alternately, the system may ignore dumps altogether (that is, autodump is set to "no") and either returns to SCM or reboots. You should set the autodump variable to "ask" or "auto" to provide a dump to DG for problem analysis. In both cases, you should append the tailored system image (usually named **/dgux**) to the dump tape. See *Notes on Tapes* below for more details.

The particular dialog depends on the type of dump device. The dump device can either be a local tape device, or, for a diskless AViiON workstation, a network device. Each of these scenarios is described below. In either case, the dialog begins as follows:

```
Do you want to take a system dump? [Y] ↵
```

Press the New Line key to answer yes, and continue with the appropriate dialog (tape or network) described below.

9.3.1 Dumping to a Local Tape Drive

You are prompted for the dump device in DG/UX common device specification format. The example below is for the common case of a SCSI tape drive on SCSI ID 4 of a workstation's integrated SCSI controller. Substitute the name of the tape device you wish to use on your system. The default device definition may not be appropriate for your system. If you have not changed the definition using **dg_sysctl**, you should enter the current device at this point. Failure to do so prevents you from taking a successful dump. Be sure you enter the correct dump device. If you enter the wrong device, the dump fails and your system will hang.

```
Dump destination device? st(insc(),4) ↵
```

You are then be prompted to mount a tape:

```
Mount tape.  Type newline when tape is ready.
```

If the system memory image is too large to fit on one tape, you are be prompted to load subsequent tape volumes:

```
Tape volume 1 completed.
Mount tape.  Type newline when tape is ready.
```

If any tape volume write fails, you can restart the dump at the beginning of that volume:

```
Hard error on tape volume 1.  Restarting volume from checkpoint.
Mount tape.  Type newline when tape is ready.
```

The final volume is rewound upon completion of the dump:

```
System dump completed successfully.
```

9.3.2 Dumping to a Network Device on a Diskless Workstation

You are prompted for the dump device in DG/UX common device specification format. The example below is for the common case of an integrated Ethernet controller on a diskless workstation. Substitute the name of the network device you wish to use on your system.

```
Dump destination device? inen() ↵
```

The system bootstrap is executed. A dump is taken before the kernel is rebooted. The dump is written to the file described as the dump target in the workstation's boot parameter entry in the server's **/etc/bootparams** file. It may also be specified in the NIS version of bootparams (Try **ypcat bootparams**). An excerpt from that file for a client named **workstation** would look like this:

```
workstation root=server:/srv/release/PRIMARY/root/workstation \
swap=server:/srv/swap/workstation \
dump=server:/srv/dump/workstation
```

Note: the dump file **/srv/dump/workstation** must exist on the server and the dump file must be exported with root access by the server or the dump will fail. An estimate of the amount of time required to create the dump file is printed on the console.

If the dump file cannot be written due to permission problems or lack of file system space, you can restart the dump by rebooting the machine from the SCM> prompt using the **-d** (dump) flag:

```
SCM> b inen() -d ↵
```

Upon completion the system is rebooted using the default system image.

9.4 Notes on Tapes

Tapes containing a system memory dump should be clearly labeled as such. Use the following format for cartridge tapes:

```
Tape 1 file 0:  memory contents (final volume) in memory dump format
Tape 1 file 1:  system image (normally /dgux) in cpio format
Tape 1 file 2:  other files, programs, etc., in cpio format
```

Do not use absolute pathnames (i.e., starting with /).

The following example shows the steps for making such a tape on a system whose primary tape drive is **/dev/rmt/0**. Use the device name appropriate for your system.

```
For file 0 on tape 1:  Dump the system memory as described above.
The final tape volume is rewound upon completion of the dump.

For file 1 on tape 1:  Use a command line like this:
```

```
# mt –f /dev/rmt/0n fsf; cd /;  echo dgux | cpio -oBcv > /dev/rmt/0n ↵
```

The tape is not rewound and is positioned for any additional files to be written to tape.

For each necessary file after file 1 on tape 1: Use a command line like this:

```
# ls filenames | cpio -oBcv > /dev/rmt/0 ↵
```

The tape rewinds after this command line.

The following example shows the steps for making a tape for a dump taken from a diskless workstation. This assumes you are making the tape while logged into the server, and that the primary tape drive is **/dev/rmt/0**.

```
# cd /srv/dump ↵
# ls diskless_client_name | cpio -oBcv >/dev/rmt/0n ↵
```

The tape is not rewound, and the kernel image may now be put on the tape.

```
# cd /srv/release/PRIMARY/root/_Kernels ↵
# ls dgux.diskless | cpio -ocvB >/dev/rmt/0 ↵
```

The tape rewinds after this command line.

You may also make the tape while logged into the workstation. This example assumes that you are using a tape drive on the workstation's server and that **/srv/dump** is mounted on your workstation.

```
# cd /srv/dump ↵
#  ls diskless_client_name | cpio -oBcv >/dev/rmt/0n ↵
```

The tape does not rewind, and the kernel image may now be put on the tape in file 1.

```
# cd / ↵
# ls dgux | cpio -ocvB >/dev/rmt/0 ↵
```

The tape rewinds after this command line.

For problems that do not involve a system dump, please put all files associated with the problem on tape file 0 in **cpio** format, using this command:

```
# ls filenames  | cpio -oBcv > /dev/rmt/0 ↵
```

Please place a label on the tape clearly indicating its contents.

End of DG/UX Release Notice Text

Index

Note: Boldfaced page numbers (e.g., **1-5**) indicate definitions of terms or other key information.