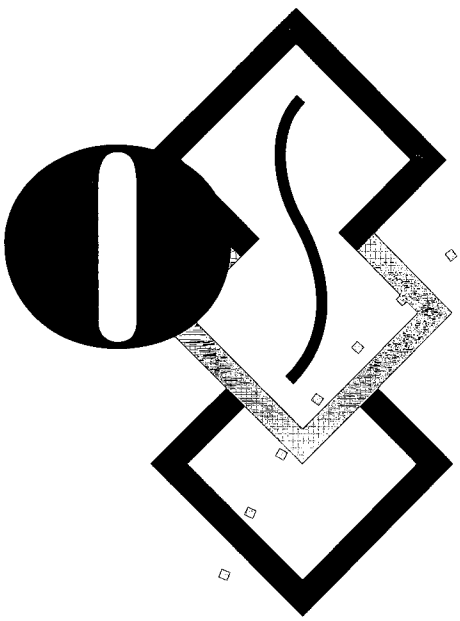


CONVEX

Managing ConvexOS: Operations Guide

Fourth Edition

CONVEX Computer Corporation
3000 Waterview Parkway
P.O. Box 833851
Richardson, TX 75083-3851
United States of America
(214)497-4000

Managing ConvexOS: Operations Guide

Order No. DSW-031

Fourth Edition
March 1994

CONVEX Press
Richardson, Texas
United States of America

Managing ConvexOS: Operations Guide

Order No. DSW-031

Printed in the United States of America

Revision information for

Managing ConvexOS: Operations Guide

Edition	Document No.	Description
Fourth	710-011830-005	Release with ConvexOS V11.0, March 1994.
Third	710-011830-003	Released with ConvexOS V10.1, July 1992.
Second	710-011830-002	Released with ConvexOS V10.0, December 1991.
First	710-011830-001	March 1991. Initial release.

Contents

Figures

Tables

Using this guide

Managing ConvexOS: Operations Guide describes how to use ConvexOS utilities to monitor and control system resources such as the line printer and tape systems. This guide is intended for system managers or their appointed assistants. Many of the tasks described in this manual do not require superuser privileges.

This book does not describe the SPU or power up and power down procedures. Refer to the *CONVEX SPU System Manager's Guide* or *CONVEX C3800 SPU System Manager's Guide,* whichever was shipped to your site.

Organization

Managing ConvexOS is a multilayered task. You require two basic types of information:

- Information to plan and allocate system resources and define their limits
- Information to monitor and control day-to-day system activity and keep the system functioning within the limits you set

For example, information required to set up the line printer system is distinctly different from the information required to manage it on a daily basis.

Because of this, *Managing ConvexOS* is packaged as a two-book set:

- *Managing ConvexOS: Configuration Guide* (hereafter referred to as the *Configuration Guide*)
- *Managing ConvexOS: Operations Guide* (hereafter referred to as the *Operations Guide*)

This volume, the *Operations Guide*, contains material on monitoring, controlling, and managing system resources, such as managing the line printer system and monitoring system resources. The system manager needs this information to maintain the system on a daily basis.

The *Configuration Guide* contains material on configuring system resources, such as setting up the disk system, customizing boot-time parameters, and setting up the line printer system. The system manager needs this information when configuring a new system or modifying the configuration of an existing system

Once you have used the *Configuration Guide* to help configure your system, you can place it on a shelf until the next time you must perform configuration tasks. All the information you need to perform daily tasks are in one volume, the *Operations Guide.*

In both books, the information is divided into tasks. For example, the *Configuration Guide* has chapters such as:

- "Setting up the disk system"
- "Setting quotas on disk space use"
- "Maintaining user accounts"

The *Operations Guide* contains chapters such as:

- "Using the operator interface"
- "Maintaining striped file systems"
- "Generating accounting reports"

When configuring the system for the first time, there is a logical order in which tasks should be performed, as some tasks require information to be in place before they can be performed. For example, users must be in the system before you can establish disk use quotas for them. Chapters in the *Configuration Guide* reflect this order, although this is not rigid in all cases.

Chapters in the *Operations Guide* are not in a specific order, because it is difficult to predict the order in which daily tasks will be performed.

The information in each chapter is also presented in the order in which it is needed. Before you can perform some of the tasks, you must plan the use of the resources being configured, and before you can plan the use of system resources, you must understand certain concepts. When this is the case, the information in each chapter is presented in the following order:

- Concepts
- Planning
- Performing

The same index appears in both guides—it is a master index that contains entries from the following books:

- *Managing ConvexOS: Configuration Guide*
- *Managing ConvexOS: Operations Guide*

Each entry in the master index is marked to indicate the book it references.

Before you begin

Full path names of commands are not given in this manual, because they change from time to time. To run commands without specifying the full path name, you must have the following directories specified in your user path:

- /bin
- /usr/bin
- /usr/local/bin
- /usr/convex
- /usr/adm
- /usr/ucb

You can use the `which` command to determine the full path name of a program or utility. Figure 1 illustrates use of the `which` command to find the full path name of the loader (`ld`) utility.

Figure 1 Using the `which` command

```
% which ld
/bin/ld
%
```

In this example, the full path name of the loader is /bin/ld.

If you use the C shell (`csh`), you can also use the `whence` command to find the program path name. The `whence` command works like `which`, but is faster.

For more information on the `which` command, refer to the which(1) man page.

Notational conventions

This section discusses notational conventions used in this book.

Command syntax

Consider this example:

```
COMMAND input_file [...] {a | b} [output_file]
   ①        ②       ③      ④        ⑤
```

1. `COMMAND` must be typed as it appears.
2. *input_file* indicates a file name that must be supplied by the user.
3. The horizontal ellipsis in brackets indicates that additional input file names may be supplied.
4. Either `a` or `b` must be supplied.
5. [*output_file*] indicates an optional file name.

General conventions

In general, the following conventions are used in this guide:

- **`Bold constant-width font`** identifies user input in examples.
- *Italics*:
  - Designate user-supplied variables in a command-line example
  - Indicate document titles
- `Constant-width font` designates input and output, including
  - Command names and options
  - System calls
  - Data structures and types
  - Directives, program statements, display examples, printout examples, and error messages returned
- Horizontal ellipsis (...) shows repetition of the preceding item(s).
- Vertical ellipsis shows that lines of code have been left out of an example.
- Words and abbreviations that indicate keyboard keys you press are identified in a distinctive bold type. For example, **RETURN** refers to the carriage return key. Words separated by

a hyphen indicate two keys that you must press simultaneously. For example, **CTRL-x** indicates that you must press and hold down the **CTRL** key and then press the **x** key.

- The word "enter" in a phrase such as "enter `ls`" means that you type the command and then press **RETURN**.
- References to ConvexOS man pages appear in the form adb(1), where the name of the man page is followed by its section number enclosed in parentheses.
- The backslash (\) character at the end of a command line indicates a continuation line follows.

Note

A Note highlights supplemental information.

Caution

A Caution highlights procedures or information necessary to avoid damage to equipment, software, or data.

Associated documents

Using this software may require information not specific to the tasks described in this document.

For more information on the ConvexOS operating system, you can order these books from CONVEX Computer Corporation:

- *ConvexOS dump and `restore` Quick Reference* (DSW-392), a quick reference for dumping and restoring file systems
- *ConvexOS Primer* (DSW-133), an introduction to ConvexOS for new users
- *ConvexOS Tape System Manager's Guide* (DSW-398), a guide and reference for ConvexOS Tape System managers
- *ConvexOS Tape System Operator's Guide* (DSW-397), a guide and reference for ConvexOS Tape System operators
- *ConvexOS Tape System Quick Reference* (DSW-391), a quick reference for the ConvexOS Tape System
- *ConvexOS Tape System User's Guide* (DSW-018), a guide and reference for the ConvexOS Tape System
- *CONVEX Guide to Attaching Multibus Peripherals (DHW-020)*
- *CONVEX VMEbus Service Kit* (DHW-050)
- *CONVEX HSP User's Guide (DHW-030)*
- *CONVEX Guide to Writing Device Drivers* (DSW-095)
- *CONVEX SPU System Manager's Guide* (DSW-022), a guide for managing CONVEX SPU OS
- *CONVEX C3800 SPU System Manager's Guide* (DSW-023), which explains procedures using the SPU OS on CONVEX C3800 Series machines
- *Managing ConvexOS: Configuration Guide* (DSW-030), a guide for configuring ConvexOS

Accessing man pages

For more information on ConvexOS, use the online man pages. To view a man page enter:

man *command*

where *command* is any valid ConvexOS command.

To print a man page, enter:

man *command* > *filename*

lpr -P<*printer*> *filename*

References made to man pages throughout this document are in the form:

cat(1)

where the man page's section number, enclosed in parentheses, follows.

Technical assistance

Hardware, software, and documentation support can be obtained through the CONVEX Technical Assistance Center (TAC):

- From locations in the continental United States
  - Customers call (800)952-0379
  - CONVEX employees call (800)952-4839
- From Canada, call 1(800)345-2384
- From all other locations, contact local CONVEX office.

Ordering documentation

To order the current edition of this or any other CONVEX document, send requests to:

CONVEX Computer Corporation
Customer Service
P.O. Box 833851
Richardson TX 75083-3851 USA

Include the order number or the exact title, as listed on the front cover.

Monitoring resources

1

Resource monitoring utilities provide you with information on system performance. With these utilities, you can monitor the use and performance of the CPU, disk, and UUCP software. This chapter discusses how to use these utilities.

Monitoring CPU use

Use the following utilities to monitor CPU use:

- `uptime`—Provides information on CPU load average on a local machine.
- `ruptime`—Provides information on CPU load average on remote machines (requires CONVEX Internet Services, an optional product).
- `ps`—Provides status information on currently running processes.
- `vmstat`—Provides information on CPU activity since the last system boot.
- `syspic`—Provides information dynamically on CPU activity on a local machine.

These utilities and how to use them are described in the following sections.

Monitoring load averages

The `uptime` and `ruptime` utilities monitor CPU load averages. The `uptime` utility monitors CPUs on the local machine, and the `ruptime` utility monitors CPUs on remote machines on a local subnet.

Monitoring local machine load averages

The `uptime` utility lists the following information for a local machine:

- Number of days, hours, and minutes the system has been up
- Number of users currently logged in
- Load average over the last 1, 5, and 15 minutes

Figure 2 illustrates example output when you execute the `uptime` command.

Figure 2 Sample output for `uptime` utility

```
5:12pm up 4 days, 11:27, 60 users, load average: 4.18, 4.40, 4.27
```

Load averages represent the number of jobs waiting in the run queue over the last 1, 5, and 15 minutes. This message tells you that the system has been up for 4 days, 11 hours, and 27 minutes; that 60 users are logged in; and that the load average over the last minute was 4.18, over the last 5 minutes was 4.40, and over the last 15 minutes was 4.27.

If you are not comfortable with the load averages seen using `uptime`, or if they jump drastically from one category to the next, you may want to run `syspic` to get more definitive information on system activity. (`syspic` is described in the subsection "Monitoring activity dynamically.")

See the uptime(1) man page for more information.

Monitoring remote machine load averages

The `ruptime` utility lists the following information for each remote machine:

- Machine name
- Whether the machine is up or down
- Number of days, hours, and minutes each machine has been up
- Number of users logged in to each machine
- Load average over the last 1, 5, and 15 minutes

The information provided by the `ruptime` utility is the same as for the `uptime` utility. The difference is that statistics are displayed for multiple machines. Figure 3 illustrates output from the `ruptime` utility.

Figure 3 Sample output for `ruptime` utility

```
pubs    up      4+02:24, 25 users, load 1.04, 1.01, 1.01
stone   up      2+20:59,  0 users, load 0.02, 0.02, 0.02
thistle down    9:06
```

If you are not comfortable with the load averages seen using `ruptime`, or if they jump drastically from one category to the next, you may want to run `syspic` to get more definitive information on system activity.

See the ruptime(1) man page for more details.

Monitoring process status

The ps utility provides basic information about running processes. Figure 4 illustrates default ps output.

Figure 4 Sample output for ps utility

```
PID     TT      STAT    TIME    COMMAND
29911   p3      I       0:00    -csh[smith]
2017    p3      R       0:00    ps
```

By default, the ps utility lists processes with the same UID as the user running ps. However, you can list processes according to other criteria. Some of the more commonly used options are

a List all processes associated with terminals.

u Produce a list sorted by CPU utilization. Each entry contains the name of the user running the process.

x Include processes not assigned to a terminal.

Use the aux options in combination to view the percentages of CPU and memory each process is using. Figure 5 illustrates output from the ps -aux command.

Figure 5 Sample output for ps -aux

```
USER    PID     %CPU    %MEM    SZ      RSS     TT    STAT  TIME    COMMAND
smith   25217   34.8    0.6     6412    308     p7    R     0:00    ps-aux
smith   22506   4.7     0.3     4912    160     p8    S     2:22    syspic
```

Each field in this output is explained below:

USER Login name of user who owns the process.

PID Identification number of the process.

%CPU Percentage of the CPU the process is using.

%MEM Percentage of memory the process is using.

SZ Virtual size of the process in 1-kbyte units.

RSS (Resident set size) Real memory size of the process in 1-kbyte units.

TT Control terminal where the process was initiated.

STAT State of the process. This can be one or more of the following letters:

V Vector processing is being used.

R Runnable process.

T Stopped process.

P Process is in page wait.

D Process is in disk or other short term wait.

S An active process (sleeping for less than 20 seconds).

I Idle process (sleeping longer than 20 seconds).

Z Zombie process (trying to exit but parent process is not waiting for it).

W Process is swapped out.

P Process is partially swapped out.

> Process is exceeding soft limit on memory.

N Process priority is reduced.

< Process priority has been artificially raised.

TIME Amount of CPU time the process has used. It might be an indication of a runaway process if this time seems excessive. Talk to the user who started the process to see if the time used seems reasonable for the process running. If it is a runaway process, kill the process or have the user who owns the process kill it.

COMMAND Command that initiated the process.

See the ps(1) man page for more details on the information that can be displayed.

Monitoring CPU activity

Both the `vmstat` and `syspic` utilities display statistics on system activity. The `vmstat` utility reports activity since the last system boot. The `syspic` utility reports current activity dynamically.

Monitoring activity since last system boot

The `vmstat` utility provides statistics to monitor system activity averaged since the last system boot, including job distribution, virtual memory load, paging and swapping activity, and disk and CPU utilization. On a multiprocessor system, the values are averaged over all CPUs. Figure 6 illustrates output from the `vmstat` utility invoked without options.

Figure 6 Sample output from `vmstat` utility

```
procs   memory  page    faults  cpu
r b w    avm     fre   re  at   pi   po fr  de sr in   sy cs  vs us sy id
1 0 0   4992   24904    0   3    5    0  0   0  0 69 165 580 75 22  4 59
```

You can specify an interval, in seconds, at which `vmstat` updates statistical information. Statistics are then averaged over the last interval rather than since system boot. For example, to view updated information every 5 seconds, enter:

```
% vmstat 5
```

Field definitions for output are provided below.

The `procs` fields provide information about the number of processes in various states. You should find few blocked (`b`) jobs. If the system is busy, however, the count of active jobs (`r`) can be large, and several of these jobs may be blocked (`b`).

`r` Number of processes in the run queue (active jobs).

`b` Number of processes blocked for resources (for example, I/O and paging).

`w` Number of processes runnable or short sleeper but partially swapped. Short sleeper is under 20 seconds.

The `memory` fields provide information about use of virtual and real memory:

`avm` Number of kbytes allocated to active virtual memory.

`fre` Number of kbytes on free list (available for allocation).

The page fields provide information about page faults and paging activity:

re Number of pages reclaimed from the swap device queue.

at Number of pages reclaimed from free pool.

pi Number of pages paged in.

po Number of pages paged out.

fr Number of pages freed.

de Anticipated short-term memory shortfall.

sr Number of pages scanned per second by the paging daemon. The paging daemon is active whenever available free physical memory is low. When the paging daemon is active, the sr field displays a nonzero value.

The faults fields provide trap/interrupt rates:

in Device interrupts per second.

sy Number of system calls per second.

cs Number of CPU context switches per second.

vs Number of vector switches per second.

The cpu fields provide information on CPU use:

us Percentage of CPU time used by users.

sy Percentage of CPU time used by the system.

id Percentage of CPU time that was idle.

Ideally, system CPU use should be as low as possible. User processes are not being run if the system CPU use is high. High system overhead can be caused by excessive context switching (cs), interrupt activity (in), or system call activity (sy). Abnormal job distribution can signify system imbalances. For example, if processes are blocked (b), the disk subsystem is overloaded or imbalanced.

System overhead increases from 60% to 70% if devices begin sending spurious interrupts or if user programs do high-speed nonbuffered I/O. Use the syspic utility to detect the port causing the problem.

If the system is heavily loaded or has little memory available, the system is forced to swap. If the runnable processes (w) shows nonzero, the system is starting to swap, which significantly reduces system performance. The best way to avoid swapping is to add more memory. However, consider administratively limiting system load if you expect to remain in a memory-poor environment. For example:

- Require users to use batch queues instead of interactive tty sessions, because the batch system can enforce limits. Additionally, swapping does not matter as much for batch jobs, because the user is not waiting for a prompt. CXbatch is an optional product.
- Add more swap partitions on different disks to increase swap bandwidth.

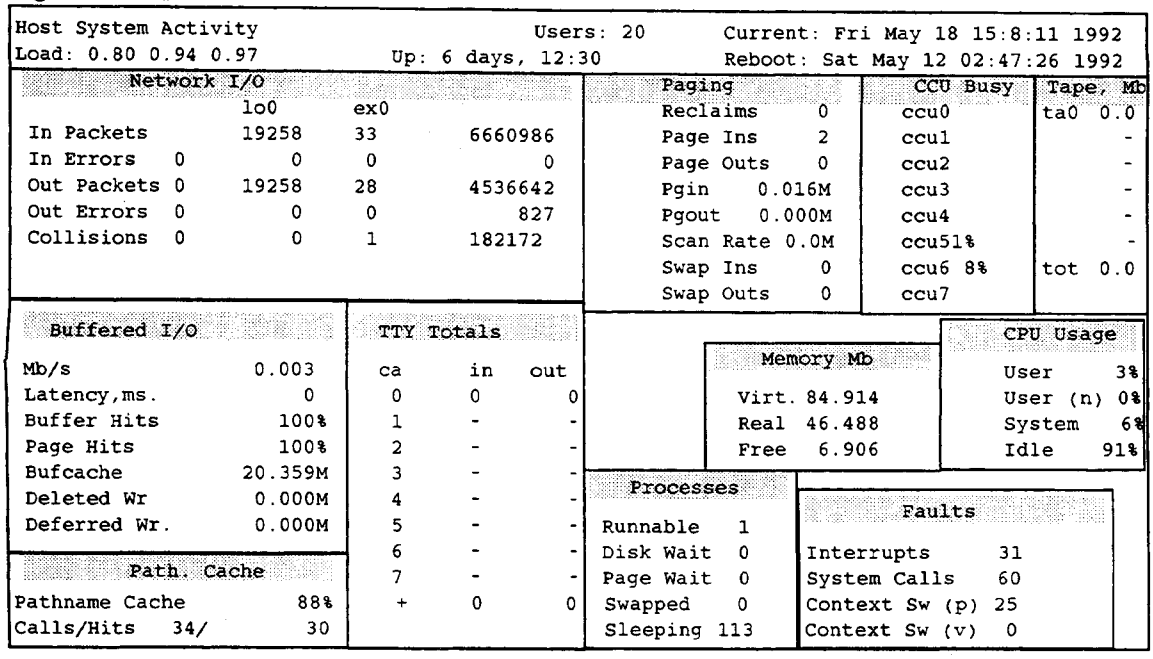
Monitoring activity dynamically

The `syspic` utility displays statistics on current system activity, such as system processes, memory, paging, faults, network use, and load average on local machines. You can use this information to determine if there are any problem areas in the system. You can view statistics on a the disk subsystem, active processes, or the network. For example, to view statistics on system activity, run `syspic` by entering:

```
% syspic
```

Figure 7 illustrates the output from this `syspic` command.

Figure 7 syspic window



The statistics are displayed in windows. The information in the windows is periodically updated. Numbers that indicate abnormal system loading are highlighted.

The first two lines on the display contain the following information:

`Users`	Number of users logged into the system.
`Current`	The last time a screen update occurred.
`Load`	Load averages representing the number of jobs waiting in the run queue over the last 1, 5, and 15 minutes.
`Up`	Elapsed time since the last reboot.
`Reboot`	Last time system was rebooted.

Each window and the information it provides is described in the following sections.

Network I/O

This window shows activity taking place on local area networks attached to the system. There are two columns of information for each device. The numbers in the first column are accrued since the last screen update. The numbers in the second column are accrued since the machine has been up. The devices shown depend on your network configuration. In Figure 7 the devices are `lo0` and `ex0`. The information displayed for `Network I/O` is

`In Packets`	Number of packets received over the network. A packet is a group of bytes transferred over a network line.
`In Errors`	Number of receive errors accrued. Receive errors occur if there is something wrong with the packet being transferred (such as an incorrect header or corrupted data). Packets generating receive errors are thrown out.
`Out Packets`	Number of packets transmitted over the network. A packet is a group of bytes transferred over a network line.
`Out Errors`	Number of transmit errors accrued. Transmit errors occur if the system receiving the data rejects the packet.
`Collisions`	Number of packet collisions accrued. Collisions occur when more than one process attempts to transmit data over the same network line. If this happens, one process must wait until the line is freed before transmitting its data.

`Collisions` should be a small percentage of the total number of `Out Packets`. If `Collisions` become excessive, you might consider getting another subnet to reduce traffic on the existing lines.

Paging

This window shows information about page faults, paging activity, and swaps. The totals are 5-second averages displayed in units per second. For example, in Figure 7, two "Page Ins" per second occurred in the last 5 seconds.

Paging is a memory management technique that allows unused portions of a program to be stored temporarily on disk to make room for more urgently needed information in main memory. The information displayed for `Paging` is

`Reclaims`	Number of pages that were paged out from a process but were given back to the process before being allocated elsewhere.
`Page Ins`	Number of faults that resulted in pages being paged in. A page fault occurs when a process requests data that is not currently in main memory.
`Page Outs`	Number of times the pager tried to page from any region.
`Pgin`	Amount of megabytes paged in, per second, as a result of faults registered in the `Page Ins` field.
`Pgout`	Amount of megabytes paged out, per second.
`Scan Rate`	Rate at which the `pageout` daemon is running through physical memory and freeing pages. As the scan rate goes up, so does the number of pages freed and number of reclaims.
`Swap Ins`	Number of processes whose swap-priority-penalty was removed. Processes are not actually swapped when this occurs, but their priority is lowered for a period of time.
`Swap Outs`	Number of processes partially swapped. These processes may still run, but their priority remains low until their swap-priority penalty is removed.

CCU busy

This window shows what percentage of the time each CCU on the system is busy. If one CCU is excessively busy while others are not used, it is an indication that too many devices are concentrated on one IOP (VIOP or HSP). Consider reallocation of devices assigned to the CCUs.

Tape, Mb/s

This window shows the transfer rates of tape drives in megabytes per second. It also provides the sum total of these transfer rates. Fields with a dash "-" at the end of the line indicate there is no tape drive connected to that line.

Buffered I/O

This window shows parameters related to buffered user I/O and the buffer cache. Buffers are used to make I/O more efficient. Instead of performing many small transfers, one larger transfer is performed each time the buffer is full. The information displayed for `Buffered I/O` is

`Mb/s`	Average amount of buffered I/O, in megabytes per second.
`Latency,ms`	Average amount of time, in milliseconds, that a request to the buffered I/O system took to complete.
`Buffer Hits`	Percentage of times a file system I/O request found the desired buffer pages already attached to a buffer header. (This value ranges between 0% and 100%.) Buffer hits cannot occur without a corresponding page hit.
`Page Hits`	Percentage of times an I/O request found the desired pages in the page cache. (This value ranges between 0% and 100%.) The page cache is a least-recently-used replaced cache containing both file system data and executable images. A page hit can occur without a corresponding buffer hit.
`Bufcache`	Effective size of the buffer cache, in megabytes. It is calculated by counting the number of pages attached to buffer headers.
`Deleted Wr.`	Amount of file system data, in megabytes, that was thrown away because the files containing it were deleted before they were written to the disk. Temporary files are often not written to the disk at all.

`Deferred Wr.` Amount of file system data, in megabytes, that was scheduled to be written to disk but had the write deferred because another request for access to the data was made while the I/O request was on the disk queue.

TTY totals

This window displays the number of characters transmitted over the tty line since the last screen update. This total is the sum of all tty lines connected to each controller. Fields with a dash (-) at the end of the line indicate there is no tty line connected to that controller. The last line contains the sum of all tty lines. The information displayed for `TTY Totals` is

`in` Number of input characters transmitted.

`out` Number of output characters transmitted.

Memory Mb

This window displays the following information, in megabytes, about use of virtual and real memory:

`Virt.` Total amount of virtual memory that might require paging to swap space. Swap space is a region on a mass storage device where processes are stored when they are swapped out. With virtual memory, the system copies infrequently used pages of memory out to disk. When a process references a memory page that is not currently in physical memory, the system automatically brings in the requested page from disk.

`Real` Total amount of memory that is in use by processes.

`Free` Total amount of memory that is not in use.

CPU usage

This window displays the percentage of usage of CPU time. Percentages are calculated as averages among all existing processors. Information is displayed for the following categories:

`User`	Percentage of CPU time used for running normal-priority user processes.
`User (n)`	Percentage of CPU time used for *running* low-priority (niced) processes.
`System`	Percentage of CPU time used for running system code.
`Idle`	Percentage of time the CPU is idle.

Path cache

This window displays information about the path name cache. The path name cache is an area in memory that contains the path name and associated inode number for recently requested files. As the path name cache fills, the oldest path names are dropped.

`Pathname`	Percentage of successful lookups `cache` made to the path name cache.
`Calls/Hits`	Number of times a path name was requested versus the number of times the path name was found in the path name cache.

Processes

This window displays the following information about processes on the system

`Runnable`	Number of processes trying to use the CPU.
`Disk Wait`	Number of processes waiting completion of disk or other short-term I/O operations, including paging and file requests.
`Page Wait`	Number of processes waiting for free memory pages, including memory for paging and file requests.
`Swapped`	Number of processes whose priority is lowered because they have *been partially* swapped.

`Sleeping`	Number of processes that are temporarily suspended. Processes sleep while waiting for I/O fulfillment, the death of an offspring, or for a specific time interval to elapse.

Faults

This window displays trap/interrupt frequency averages per second over the last 5 seconds. A trap is a machine instruction used as the system call interface between application software and the kernel. When an application issues a trap instruction, the kernel takes over. The kernel performs the requested service and then resumes the application program.

Interrupts indicate a condition such as the arrival of data at an interface, the expiration of a timer, or the completion of a data transfer. Most interrupts are generated by peripheral devices. When an interrupt is activated, the current task is suspended while the instructions in the interrupt handler are performed. When the interrupt servicing is complete, the system resumes the original task. The major role of interrupts is to enable the system to supervise data transfers.

`Interrupts`	Number of interrupts, per second, not including the clock.
`System Calls`	Number of system calls, per second.
`Context Sw (p)`	Context switch rate, per second, for processes.
`Context Sw (v)`	Context switch rate, per second, for vectors.

Monitoring processes dynamically

For more information about processes, use the `syspic` utility with the `-p proc` argument. For example, enter

```
% syspic -p proc
```

Figure 8 illustrates output for this command.

Figure 8 `syspic -p proc` output

```
Host System Activity                    Users: 20      Current: Fri May 18 15:8:11 1992
Load: 0.80 0.94 0.97                 Up: 6 days, 12:30      Reboot: Sat May 12 02:47:26 1992

        Memory Mb                    Per-CPU Usage
                                                 total    cpu0     cpu1
Virt.     85.898                     User        24%      25%      24%
Real      47.902                     User (n)    17%      19%      14%
Free       5.609                     System      41%      40%      41%
                                     Idle        18%      17%      20%

USER       PID      %CPU     %MEM     SIZE     RSS      TT       STAT  TIME     COMMAND
smith      19909    20.9     1.9      7028     1056     p7       R     0:41     syspic -p proc
jones      19918    3.4      0.8      480      412      pe       S     0.00     mail
root       2266     2.7      0.8      1152     408      ?        S     1:07     xterm -sf -d star:0
root       19916    1.9      0.3      184      140      ?        S     0:00     rstatd
jones      2299     0.7      0.6      404      344      pe       S     0:09     -tcsh
root       241      0.3      0.1      48       24       ?        S     34:08    /etc/update
brown      3405     0.2      0.2      692      112      p0       S     1:06     uxwindows
root       101      0.2      0.0      116      4        ?        S     51:23    /etc/nfsd 4
root       103      0.2      0.0      116      4        ?        S     51:15    /etc/nfsd 4
root       24533    2.9      0.9      1148     456      ?        S     0:30     xterm -sf -d moon:0
```

The statistics are displayed in windows. The information in the windows is periodically updated. Numbers that indicate abnormal system loading are highlighted.

The first two lines on the display contain the following information:

`Users`	Number of users logged into the system.
`Current`	The last time a screen update occurred.
`Load`	Load averages representing the number of jobs waiting in the run queue over the last 1, 5, and 15 minutes.
`Up`	Elapsed time since the last reboot.
`Reboot`	Last time system was rebooted.

Each window and the information it provides is described in the following sections.

Memory Mb

This window displays information, in megabytes, about use of virtual and real memory.

`Virt.` The total amount of virtual memory that might require paging to swap space. Swap space is a region on a mass storage device where processes are stored when they are swapped out. With virtual memory, the system copies infrequently used pages of memory out to disk. When a process references a memory page that is not currently in physical memory, the system automatically brings in the requested page from disk.

`Real` The total amount of memory that is in use by processes.

`Free` The total amount of memory that is not in use.

Per-CPU usage

This window displays the percentage of usage of time as an average for each CPU in the following categories:

`User` Percentage of CPU time used for running normal-priority user processes.

`User (n)` Percentage of CPU time used for running low-priority (niced) processes.

`System` Percentage of CPU time used for running system code.

`Idle` Percentage of time the CPU is idle.

Additional information is provided about the 10 processes using the most CPU resources

`User` Login name of user who owns the process.

`PID` Identification number for the process.

`%CPU` Percentage of CPU the process is using.

`%MEM` Percentage of real memory the process is using.

`SIZE` Virtual size of the process in 1-kbyte units.

`RSS` Resident set size. Real memory size of the process in 1-kbyte units.

`TT` Control terminal where the process was initiated.

`STAT` Status of process. This can be one of the following:

- `V` Vector processing is being used.
- `R` Runnable process.
- `T` Stopped process.

P Process is in page wait.

D Process is in disk or other short term wait.

S An active process (sleeping for less than 20 seconds).

I Idle process (sleeping longer than 20 seconds).

Z Zombie process (trying to exit but parent process is not waiting for it).

W Process is swapped out.

P Process is partially swapped out.

> Process is exceeding soft limit on memory.

N Process priority is reduced.

< Process priority has been artificially raised.

TIME Amount of CPU time the process has used. It might be an indication of a runaway process if this time seems excessive. Talk to the user who started the process to see if the time used seems reasonable for the process running. If it is a runaway process, kill the process or have the user who owns the process kill it.

COMMAND Command that initiated the process.

See the syspic(8) man page for more details on what you can view.

Monitoring disk use

The df, du, and quota utilities monitor disk use:

- The df utility reports the amount of free disk space, in kilobytes, available in a file system.
- The du utility reports the number of kilobytes used by files and directory subtrees.
- The quota utility displays a user's current disk usage and limits.

This section describes how to use these utilities.

Monitoring free disk space

If you specify a file system with the df command, df reports free space for the specified file system. If you specify a file with the df command, df reports on the free space for the file system where the specified file resides. If you do not specify a file or a file system, the df utility reports on all mounted file systems. For example, enter the following command to report on free space for the /mnt file system:

```
% df /mnt
```

Figure 9 illustrates sample output from this command.

Figure 9 Sample output from df utility

```
File systemKbytesused avail   capacity         Mounted on
/dev/st6163423 141277 5803    96%    /mnt
```

The total amount of disk space is reported in the Kbytes column. Entries in the used and avail columns totaled together will show 10% less than the total number in the Kbytes column. ConvexOS attempts to keep 10% of the disk unallocated to prevent disk fragmentation. See the df(1) man page for more details on this utility.

Monitoring used disk space

The du utility counts the actual number of disk blocks used, which is different from the ls command, which reports on the total file system size.

You can specify either a file or directory name with the du command. If you specify a file, you receive the number of kilobytes used by the specified file. (You must use the -a option to get file totals.)

If you specify a directory name, you receive totals for each directory within the specified directory, recursively. If you specify the -a option, you also get totals for the files within the specified directory, recursively.

If you do not specify a file or directory name, the du utility assumes the top of the file system for the current working directory.

To generate disk use totals for the tmp directory in user smith's home directory, enter

```
% du -smith/tmp
```

Figure 10 illustrates sample output from this command.

Figure 10 Sample output from du utility

```
770     /mnt/smith/tmp/oper
1425    /mnt/smith/tmp/config
133     /mnt/smith/tmp/source/sysfiles
64      /mnt/smith/tmp/source/opfiles
599     /mnt/smith/tmp/source
2883    /mnt/smith/tmp
```

See the du(1) man page for more details on this utility.

Monitoring current use and limits

If your system uses quotas to control the amount of disk space used by users, use the `quota` command to display a user's current disk usage and limits. For example, to check the quota use for user smith, enter

```
% /usr/ucb/quota -v smith
```

Figure 11 illustrates sample output from this command.

Figure 11 Sample output from the `quota` utility

```
Disk quotas for smith (uid 2500):
Filesystemusage quota   limit   timeleftfilesquotalimittimeleft
/mnt    17      100     150             13      0       0
```

For more information on the `quota` utility, refer to the quota(1) man page.

Monitoring UUCP use

The uusnap, uulook, and uuclean utilities monitor UUCP activity and remove old UUCP files. This section describes how to use each of these utilities.

Monitoring pending UUCP activity

The uusnap utility generates a table showing the status of pending UUCP activity. Figure 12 illustrates sample output from this command.

Figure 12 Sample output from uusnap utility

```
smu      1 Cmd   ---     ---     TALKING
allegra  1 Cmd   ---     ---
ctvax    1 Cmd   ---     ---     DIAL FAILED Retry time 53 mins
cua1     ____    ____    ____    LOCKED
```

The columns describe, from left to right

- System or resource name
- Number of outgoing commands
- Number of incoming or outgoing data files
- Number of commands queued for local execution
- Status message

The LOCKED status shown in row four indicates that the resource is being used and is unavailable for acquisition by another process. The status message can include the time remaining before UUCP can retry the call and the number of times UUCP has unsuccessfully attempted to reach the site.

For more information on the uusnap utility, refer to the uusnap(8) man page.

Monitoring the UUCP logfile

The uulook utility allows you to view the end of the UUCP logfile and monitor it, displaying new entries as they occur, until you send an interrupt (usually CTRL-C). Figure 13 illustrates sample output for the uulook utility.

Figure 13 Sample output from uulook utility

```
smith rice (2/15-12:41-985) SUCCEEDED (call to convex)
smith rice (2/15-12:42-985) OK (startup)
jones info (4/1-14:28-9574) FAILED HANDSHAKE (You are unknown)
```

The columns describe from left to right

- Name of the user calling uucp
- Name of the system called
- Time and date of the call and the associated process ID
- Call status
- Status detail

See the uulook(8) man page for more details on this utility.

Removing old UUCP files

The uuclean utility removes old files from the /usr/spool/uucp subdirectories. Files that are removed are usually the result of interrupted uucp queuing or transmission procedures. The uuclean utility, typically run by cron, deletes all files in the spool directories older than a specified number of hours. Figure 14 illustrates a typical script run by cron.

Figure 14 Sample uuclean cron script

```
/usr/lib/uucp/uuclean -d/usr/spool/uucp/D.convex -n99 -pD.convex
/usr/lib/uucp/uuclean -d/usr/spool/uucp/C. -n169 -pC.
/usr/lib/uucp/uuclean -d/usr/spool/uucp/D. -n169 -pD.
```

Each line in the script begins with an invocation of uuclean. The -d option specifies the name of the subdirectory to be cleaned. The -n option specifies the number of hours old the file must be to be removed. The -p option specifies the prefix the file must match to be removed.

Refer to the uuclean(8) man page for more details on this command. Refer to Chapter 3, "Controlling processes," on page 29, for information on the cron utility.

Using the operator interface 2

Because superuser privileges provide access to every command and file in the system, it is not desirable to grant superuser privileges to everyone who may need to perform a maintenance task. The operator interface, commonly called op, allows you to grant restricted access to superuser commands without granting superuser privileges. op provides an operator class of access to superuser commands. This chapter describes how to use the operator interface.

The operator interface system

ConvexOS distinguishes two classes of users: ordinary users and superusers.

A superuser has privileged access to the computer system. The superuser can perform any operation and has full read, write, and execute privileges for all files, regardless of who owns them or their access permissions. Ordinary users have access only to their files and other files to which they are explicitly granted access.

With the operator interface, the system manager can establish additional classes of users. The users in these classes are granted access to a set of commands that typically require superuser privileges; they do not receive full superuser privileges. As system manager, you can restrict

- **Who may be an operator**—You can restrict an operator class to an individual or several individual users, a group of users as defined in the /etc/group file, or a combination of individuals and groups.
- **What commands the operator may use**—You can restrict the specific commands executed by users in an operator class. These commands can include custom scripts or programs designed for your site.
- **What arguments the operator may pass to the command**—You can restrict the arguments that may be passed to these commands.

Each command an operator can perform is given a task name. For example, the system manager might name a task `weekly` to describe a command he or she wants performed on a weekly basis.

Besides defining what command is executed for each task, the system manager can specify arguments the operator can pass to the command. This can be either a literal or variable argument.

Literal arguments specify command arguments, such as `0Gun` for a `dump` command, or specific file systems, such as /mnt. Variable arguments allow an operator to enter a desired value when the task is invoked. However, the system manager can specify limitations on what values an operator can enter for variable arguments as well.

See the op(8) and op.access(5) man pages for more details. See the chapter "Granting Operator Class Privileges" in the *Configuration Guide* for setup information.

Using the op help facility

Anyone designated as an operator can use the op help facility to determine which tasks are available to them or to determine the arguments available for a task. If a user enters the following command

```
% /etc/op -h
```

the op facility lists the functions available to that user. For example, user smith has operator access to two tasks: `weekly` and `daily`. If user smith executes the op help facility, op generates the following list:

```
weekly

daily
```

User jones does not have operator access. If user jones executes the op help facility, op generates the message

```
jones is not allowed to execute any mnemonics
```

To get information on available arguments for one of these tasks, for example on a task called `weekly`, enter the following command:

```
% /etc/op -h weekly
```

If the operator has access to a task that has one variable argument specified in the command syntax, and that variable can be replaced with either /, /usr, /mnt, or /project, op displays the following kind of information, as illustrated in Figure 15.

Figure 15 Sample `op -h` output

```
% op -h weekly
Help for mnemonic "weekly"
          op weekly <arg1>
where
          <arg1> is "/" OR "/usr" OR "/mnt" OR "/project"
```

If no limitations are specified, you can enter any valid value for the argument. Unless an argument is specified as optional, the operator must supply a corresponding value for any variable argument when they invoke the task.

Invoking op tasks

Anyone with operator access to any op task can invoke that task using the op command. The format of the op command is

```
% /etc/op task_name [arg...]
```

where

task_name is the name of the task as defined in the op.access file. Each task calls an executable command, utility, or script.

arg is any argument the task will accept. For information on how to determine valid values for tasks, refer to the section, "Using the op help facility," on page 27.

For example, to invoke a task called `weekly` that allows you to back up either /usr, /mnt, or /project, you can enter

```
% /etc/op weekly /usr
```

Controlling processes

3

ConvexOS process control utilities allow you to

- Schedule processes in the future
- Remove jobs from the cron execution file
- Change execution priorities

This chapter describes how to use these utilities.

Changing process priorities

When a file is executed, it becomes a process. Each process has an execution priority associated with it based on the nice value of the process. The default nice value for a program run from a shell is zero.

You can alter a process's execution priority by altering the process's nice value. This can be done either when the process is invoked, using the `nice` utility, or after the process is invoked, using the `renice` utility. This section describes how to use the `nice` and `renice` utilities.

You can set the defaults for the `renice` utility by setting the `auto_nice_factor` boot-time parameter. Refer to Chapter 15, "Customizing kernel boot-time parameters," of the *Configuration Guide* for details on setting this parameter.

Specifying nice values

Use the `nice` utility when invoking a process to specify the process's nice value. The nice value is used to determine the process's execution priority.

The `nice` utility accepts as an argument a positive integer between 1 and 64, which increments the existing nice value. A higher number means a lower priority. That is, a process with a nice value of 5 runs before a process with a nice value of 10.

The superuser can increase the priority of a process between negative 1 and 64 using the `nice` utility by specifying a negative numerical argument. This will make the process run before other programs so it will take less time to finish.

For example, if a process myprogram has a nice value of 0, and you execute the following command, myprogram starts with a nice value of 3.

```
% nice +3 myprogram
```

If you do not specify a positive integer between 1 and 64, the `nice` utility increments the default process priority by 4.

Note

Raising the priority to 3 or 4 causes jobs to run somewhat slower. Processes with a priority set at 64 run only when no other processes are running.

Changing nice values

Once a process is started, the owner of the process can affect its execution priority by changing its nice value using the `renice` utility. You can change the nice value for a single process, all processes in a process group using the `g` option, all processes owned by a user with the `u` option, or any combination of these.

You need the process ID (PID) number of a process to change the nice value of a running process. You can get the PID number using the `ps` command. For example, if you know you have a program called myprogram running and you want to find its PID, enter

```
% ps
```

You receive output similar to that shown in Figure 16.

Figure 16 Sample output for `ps` command

```
PID        TT        STAT      TIME        COMMAND
29911      p3        I         0:00        -csh[smith]
2017       p7        R         0:00        myprogram
```

The first column in the `ps` command output provides the PID number for myprogram. In this example, the PID is 2017.

Next, execute the `renice` command to change the process's nice value. For example, the following command sets the nice value to 3 for the process numbered 2017:

```
% /etc/renice +3 2017
```

The `renice` utility accepts as an argument a positive integer between 1 and 64, which changes the existing nice value. A higher number means a lower priority. That is, a process with a nice value of 5 runs before a process with a nice value of 10.

The superuser can specify any positive or negative value between -64 and +64. However, if you make the nice value very negative, the process cannot be interrupted. To regain control, make the nice value greater than zero.

Users other than the superuser can only alter the nice value of processes they own and only by a positive value. Users cannot increase nice values of their own processes even if they decreased the priorities in the first place.

Scheduling processes

The at utility executes specified commands once at a specified time and date. The cron utility executes specified commands either once or several times at specified times and dates. If you only want the process to run once, use the at utility. cron continues to run the command as long as the specified date matches the current date.This section describes how to use the at and cron utilities.

Scheduling future one-time execution

Use the at utility to schedule automatic execution of a shell script or program at a specified time and date. The time can be from one to four digits long. One- and two-digit times are assumed to be hours. Three- and four-digit times are assumed to be hours and minutes. For example, 435 is interpreted as 4:35 a.m.

You can optionally include an a for a.m., p for p.m., n for noon, or m for midnight with the time specification. For example, 1200n is interpreted as noon. If no letters follow the time digits, a 24-hour clock time is used.

You can optionally include a day of the month or day of the week. Specify a day of the month by including a month name followed by a day number. For example, specify jan 24 to schedule execution on January 24.

Specify a day of the week instead of a date by simply specifying the day name. Add week after the day name to schedule a week ahead. For example, specify fr week to schedule execution on Friday of next week. If you just specify week without a day name, execution is seven days from the current date.

If a day is not specified, the current day is assumed. Names of months and days can be truncated.

For example, to schedule execution of a program called big.program at 4:00 a.m. on January 24, enter

```
% at 4am jan 24 big.program
```

The at utility copies the specified script to the /usr/spool/at directory as a process file. When the time specification is met, the atrun utility executes the process file. The atrun utility is normally executed every 15 minutes by the cron utility.

You can create a shell script of commands to execute using the `at` utility. For example, if you want to execute the `ls` command at 4:00 a.m. on the current day, enter the following commands:

```
% at -C ls -f ls.script

% at 4am ls.script
```

The first command specifies the command you want executed (`ls`) and places it in an output file called ls.script. The second command schedules the ls.script file to execute at 4:00 a.m. on the current day. The `-C` option specifies that there is only a single command to execute.

If you want to schedule multiple commands to execute at a specified time without first placing them in a shell script, use the `at` command without specifying the `-C` option or an input file name. The `at` utility prompts for commands from standard input until you press **CTRL-d**. Figure 17 illustrates a sample interactive at session.

Figure 17 Sample interactive `at` session

```
% at 4am
at> cd ..
at> ls -l
at> <EOT>
```

The `at` utility copies the specified script to the /usr/spool/at directory as a process file. When the time specification is met, the `cron` utility executes the process file.

If the `-m` flag is specified, mail is sent to the user after the job has been run. If errors occur during execution of the job, then a copy of the error diagnostics is sent to the user. If no errors occur, then a short message is sent informing the user that no errors occurred.

The `-f` *outputfile* option may be used to generate a shell script file just like one that would be run by `atrun`, placing the output in the named outputfile rather than in the directory of files to be run automatically. This option is useful because it creates a shell script that takes care of restoring the current environment setting, but leaves the user with complete freedom to run the script.

See the at(1) man page for more details on this command.

Scheduling multiple-time executions

`cron` executes commands at specified dates and times according to the instructions found in the /usr/lib/crontab file or any .crontab file residing in a user's home directory. Use the `cron` utility to automatically execute commands, at regular intervals. For example, you can use `cron` to

- Truncate growing files on a regular basis
- Search for and delete out-of-date messages
- Remove editor checkpoint files
- Print system statistics
- Start up UUCP for regular calls

Each line of the crontab file represents one activity; fields in each line are separated by spaces or tabs. The first five fields in a crontab entry are integers that specify when the command should be performed. The format is

minute hour date month day command

where

minute	is any number between 0 and 59, inclusive.
hour	is any number between 0 and 23, inclusive.
date	is any number between 1 and 31, inclusive.
month	is any number between 1 and 12, inclusive.
day	is any number between 1 and 7, inclusive. One (1) indicates Monday, two (2) indicates Tuesday, three (3) indicates Wednesday, and so on.
command	is the command that is executed when the time element is met. A percent character in this field is translated as a newline character.

Each of the first five fields can be a single value or a list of values separated by commas. Use an asterisk to specify all legal values. To specify an inclusive range, separate two numbers with a minus sign. Figure 18 illustrates a sample /usr/lib/crontab file.

Figure 18 Sample /usr/lib/crontab file

```
0 1 * * * yourprogram
0 9-17 * * 1-5 myprogram
```

The first line in this example runs `yourprogram` at minute 0 of hour 1 (1:00 a.m.) every day, specified by the asterisk in the day, month, and day of week fields. The second line runs `myprogram` at minute 0 of hours 9 through 17 (9:00 a.m. until 5:00 p.m.) on Monday through Friday, every week of the month.

The `cron` utility produces a variety of messages about syntax errors, file-access errors, and illegal use errors. Messages are handled by `syslogd` or mailed to the user. Commands are not executed if they generate an error message.

`cron` updates its task list once an hour. To force an update, use the `tellcron` utility. This informs you of any syntax errors immediately.

See the cron(1) and crontab(5) man pages for more details on the `cron` utility.

`cron`

The `cron` system has been modified to incorporate some POSIX 1003.2 functionality. crontab files are now spooled to /usr/spool/cron as *<username>*.tab and any cronrc files are spooled as /usr/spool/cron/*<username>*.rc. To preserve backward compatibility, a new `autocron` daemon has been written that will update the spooled crontabs every hour.

If the /etc/posixcron file exists, and cron is not started with the `-a` flag, `cron` runs in POSIX mode and crontab files are spooled through the crontab interface.

If `cron` is started with the `-a` flag or the /etc/posixcron file does not exist, `cron` starts the `autocron` daemon and backwards compatibility is preserved. The crontab interface can still be used.

`tellcron` has also been modified to provide POSIX 1003.2 functionality. There are several options that can be used to perform operations on your crontab file.

For more information, refer to the tellcron(1), crontab(5), cron(8), and autocron(8) man pages.

Managing the line printer system

4

The line printer system is a collection of programs and files that manage printer operations. This includes setting up the spool system, handling print requests, managing queues, and enabling and disabling printers. This chapter describes how the line printer system works and introduces programs you will use to maintain the line printer system on a day-to-day basis.

Instructions for configuring new printers are found in the following chapters:

- Chapter 2, "Adding devices," on page 21 in the *Configuration Guide*, describes operating system files you need to modify to add a new printer to the system.
- Chapter 5, "Setting up the line printer system," on page 119 in the *Configuration Guide*, describes procedures for configuring printers and creating spool directories.

Error messages are listed in Appendix A, "Line printer system error messages," in this guide.

Refer to the *ConvexOS Primer* for information on using the line printer system and submitting print jobs with the `lpr` command.

The line printer daemon

ConvexOS handles many tasks in the system by using daemons. A daemon is a system process, usually started at boot time, that handles repetitive system functions in the background. A daemon usually lives as long as the system is running and waits between requests.

The line printer daemon is called `lpd`; it is started from the rc.std script at boot time and takes care of numerous tasks in the line printer system. When `lpd` is invoked, it scans the /etc/printcap file to determine what printers are on the system and prints files that were left in the queues when the system went down. After it has taken care of these tasks, `lpd` is ready to process new requests.

The line printer daemon handles requests to

- Transfer files to the spooling area
- Transfer files between queues
- Display a printer queue
- Remove files from a queue
- Print files in the queue

`lpd` uses sockets to handle requests. When a request is received, `lpd` forks a child process to handle the request so the parent `lpd` process can listen for additional requests. Consequently, there may be several copies of `lpd` running at the same time.

When a user submits a print request with the `lpr` command, the file is first copied into a dedicated spooling directory within the /usr/spool directory. For example, files sent to the default line printer might be queued in the directory /usr/spool/lpd. From here, the file is processed by `lpd` and printed on a local printer or transferred to a remote host for printing on a remote printer.

To watch `lpd` work, look in /usr/spool for a file called lpd.lock and for the lpd directory. The /usr/spool directory is the default spool directory. The actual spool directory for a given printer is specified in the sd field of the /etc/printcap file. The lpd directory contains three files called .seq, lock, and status. You can list the lpd directory contents with the `ll -a` command.

The .seq file contains the queue sequence number of the next job to be queued. The lock file contains two lines: the PID of the daemon assigned to the line printer and the job ID of the job currently printing (or the last one printed). The status file contains a line describing the status of the printer (for example, active and printing, needs paper). This line is written into the status file by a daemon that communicates with the line printer device driver, or by the `lpc down` command (see the next section, "Managing printers"). The `lpq` command reads the status file to determine the status of the line printer queue.

lpd.lock is a lock file used by the master printer spooler to hold the PID of the `lpd` process. The master spooler coordinates all print requests and directs jobs among printers attached to the system.

Managing printers

The `lpc` (line printer control) utility is the primary tool for managing the line printer system. It allows you to control and maintain the operation of printers described in the /etc/printcap file. You can use the `lpc` utility to

- Start or stop a printer
- Redirect a print queue
- Enable or disable print queues
- Rearrange the order of jobs in a queue
- Display the status of a printer, its daemon, and its queue

Some `lpc` commands require root privileges. The format for the `lpc` utility is

`lpc` [*command* [*parameter* ...]]

where

command is an `lpc` command. This can be one of the following:

`abort`	Disable a printer immediately.
`clean`	Remove files in a printer queue.
`disable`	Disable a printer queue.
`down`	Disable printing and queuing; put message in status file.
`enable`	Enable a printer queue.
`exit`	Exit the `lpc` utility.
`help`	Provide help on available commands.
`redirect`	Redirect jobs from one printer to another.
`restart`	Restart a printer daemon.
`start`	Enable printing on a printer.
`status`	Display printer status.
`stop`	Disable printing on a printer.
`quit`	Exit the `lpc` utility.
`topq`	Place a job at the top of the print queue.
`undirect`	Remove redirection from a printer.
`up`	Enable printing and queuing; remove message from status file.
`?`	Provide help on available commands.

parameter is a parameter or list of parameters for the command. Typically, this is the name of a printer or `all` to specify all printers. If no parameter is given, most `lpc` commands return a usage message such as

```
Usage: restart {all | printer ...}
```

If `lpc` is invoked without arguments, it runs interactively and displays the following prompt:

```
lpc>
```

When entering commands, you can specify the first few letters of the command instead of the entire command name. The `lpc` utility recognizes abbreviations, providing you supply enough letters to distinguish one command from another. This works in both interactive and noninteractive modes. For example, the `clean` command may be shortened to `c` while the `status` command may only be abbreviated to `stat` in order to distinguish it from the `start` command. If you enter an abbreviation that is ambiguous, `lpc` responds with the message

```
?Ambiguous command
```

You can use either the `exit` command or the `quit` command to exit the `lpc` utility. The end of file character, **CTRL-d**, is also recognized as a command to quit `lpc`.

Getting help

The `lpc` utility provides online help with the `help` or `?` command. The `help` command gives a short description of each command in the argument list. For example

```
lpc> help status
```

displays a one-line description of the `status` command

```
status          show status of daemon
```

If no arguments are given with the `help` command, the list of valid commands is displayed. Use the `help` command whenever you are uncertain about the function of an `lpc` command.

Starting and stopping printer daemons

The `lpc start` and `stop` commands control the line printer daemon, `lpd`. To stop a printer daemon, execute the `lpc stop` command. When you issue a `stop` command, `lpc` waits for the current job (if any) to finish printing. When the job is complete, the daemon for that printer is stopped. An example of the `stop` command is

```
lpc> stop myprinter
```

You can also issue the `stop` command to all printers by entering

```
lpc> stop all
```

The `stop` command does not disable queuing of print jobs. Users can submit new jobs to a stopped printer; the jobs wait in the queue until a daemon is started.

To start the printer daemon again, use the `start` command

```
lpc> start myprinter
```

or

```
lpc> start all
```

The `lpc abort` command also disables printing to any or all printers. It differs from the `stop` command by not waiting for the current job to finish printing before stopping the printer. When issued, the `abort` command terminates the printer daemon immediately. The `abort` command does not disable queuing of print jobs.

Restarting a printer

One of the most common uses of `lpc` is to restart a printer daemon. When a printer daemon dies, jobs pile up in the queue and nothing is printed. To restart the printer, use the `restart` command as shown below

```
lpc> restart myprinter
```

You can use several methods to determine whether a printer has lost its daemon

- The `lpc status` command shows the status of daemons and queues on the local machine.
- You can exit `lpc` and use the `lpq` command from the shell to examine printer queues. The output from `lpq` shows you all pending print jobs and notifies you of any error conditions with the printer or the printer daemon.
- You can use the `ps` and `grep` commands as shown in Figure 19.

Figure 19 Checking for an active `lpd` process using `ps` and `grep`

```
% ps ax | grep lpd | grep -v grep
303 ?    I      0:44 /usr/lib/lpd -f /etc/hosts.equiv
```

If no `lpd` process is displayed, restart the daemon. If an `lpd` process is displayed, a printer daemon exists, and a restart is not necessary.

Enabling and disabling queues

The `enable` and `disable` commands control printer queues. When a `disable` command is issued, the printer queue is turned off and no new jobs are queued. However, any job in the queue is printed. If anyone attempts to submit new jobs to the printer using `lpr`, the following warning message is displayed:

```
lpr: Printer queue is disabled
```

The `enable` command reverses the effect of the `disable` command.

Console printer

If the console printer is enabled but the printer itself is offline, the system will crash. The SPU has a limited number of message buffers to the console and if the printer is enabled and offline, the buffers cannot be emptied. When the buffers become full, no messages can be received, which causes the system to crash.

Redirecting queues

The `lpc redirect` command allows you to redirect print jobs from one printer queue to another. The format of the `redirect` command is

```
redirect [all] source destination
```

When you issue the `redirect` command, jobs submitted to the *source* printer are redirected to the *destination* printer.

If you specify `all` in the command line, `lpc` will tell the machine hosting the printer to do the `redirect`. This option redirects all jobs submitted from anywhere on the network. This means all jobs submitted will be redirected. The hosting machine must be a Convex to support this functionality.

If no option is supplied, only jobs submitted from the machine where `redirect` was issued are affected.

Use the `undirect` command to remove redirection from a printer. The format of this command is

```
undirect [all] printer
```

printer is the source printer specified in the `redirect` command.

Manipulating jobs in the queue

The contents of a printer queue can be adjusted using the `topq` and `clean` commands. The `topq` command moves print jobs to the top of the queue, causing those jobs to print first. You can move a job by its job number or by entering the user name of the job's owner. In the latter case, all print jobs owned by the user in the specified queue are moved to the top of the queue. For example, the command

```
lpc> topq myprinter 123
```

would place job number 123 at the top of the queue. The following example specifies a user name:

```
lpc> topq myprinter jones
```

The `clean` command deletes all files from the printer queue that begin with the letters cf, tf, or df. This command also removes any files remaining from failed jobs.

Scheduling downtime

When you need to disable a printer for things such as scheduled maintenance, use the `down` command. This command is similar to the `disable` command, except that it allows you to enter a message that is displayed when the `lpq` or `lpc status` commands are used to examine queue information. The message you enter is written to the status file in the printer's spool directory and remains until you enable the queue with the `lpc enable` command.

An example of the `lpc down` command is

```
lpc> down myprinter Down for maintenance.
```

To enable the printer, use the `up` command. The `up` command reverses the effect of the `down` command, enabling the printer and removing the message from the status file. An example of the `up` command is

```
lpc> up myprinter
```

Managing queues

In addition to the `lpc` commands that allow you to examine and adjust information in printer queues, three other utilities can help in managing queues

- `lpq`
- `lpmv`
- `lprm`

These utilities are not part of the `lpc` utility; they are invoked directly from the shell. Each is described below.

Checking printer queues

You can determine status of a queue and display the list of jobs waiting to be printed using the `lpq` utility. The status typically reported by `lpq` includes whether the daemon is active or disabled, and whether the printer is currently printing, out of paper, or idle.

The format of `lpq` is

`lpq [+[`*n*`]] [-f] [-l] [-P`*printer*`] [`*job#*`] [`*user*`]`

where

`+`	displays the queue until it is empty. If you supply a number immediately after the +, `lpq` sleeps for that many seconds between scans of the queue.
n	specifies the number of seconds `lpq` should sleep between queue scans (refer to the + option).
`-f`	displays the queue forever. When this option is specified, all other arguments are interpreted as user names or job numbers. This option overrides `+[`*n*`]`.
`-l`	displays information about each file in the print job.
`-P`*printer*	specifies the print queue. If the `-P` option is not specified, the default print queue is displayed or the queue specified by the PRINTER environment variable is used (providing this variable is set).
job#	displays a specific job in the queue identified by its job number.
user	displays all jobs for a user.

If no arguments are supplied, `lpq` displays the default queue or the queue specified by the PRINTER environment variable (providing this variable is set).

An example printer-queue status request using the `lpq` utility is shown in Figure 20.

Figure 20 Example `lpq` output

```
% lpq -Pmyprinter
myprinter is ready and printing
Rank    Owner      Job   Files                 Total Size
active  smith      322   testplan              2896734 bytes
1st     jones      92    chap02                3469 bytes
```

Moving print jobs between queues

The `lpmv` utility moves jobs from one printer queue to another. Because spool directories are protected from normal users, `lpmv` is the only way users can move their jobs between queues.

The format of `lpmv` is

`lpmv [-a] [-P`*printer*`] [`*user*`] [`*job#*`] `*destination*

where

`-a`	moves all jobs owned by user invoking `lpmv`.
`-P`*printer*	specifies the source print queue. If the `-P` option is not specified, the default printer queue is used.
user	moves all jobs for a user. This option is useful for system managers to move a user's files all at once. root can specify multiple users.
job#	moves a specific job in the queue identified by its job number.
destination	is the name of the printer that you want to transfer jobs to.

If no arguments are supplied, `lpmv` moves and restarts the active job (if owned by the user) to the destination printer. The `lpmv` utility displays the names of files it moves. Nothing is displayed if a match is not found. When `lpmv` moves a file, it may kill an active daemon. If the daemon is killed, `lpmv` automatically restarts a new daemon after moving the specified files.

An example of an `lpmv` request is shown in Figure 21. Note that `lpmv` tells you the new job number when the move is complete.

Figure 21 Example of lpmv output

```
% lpmv -Pmyprinter 456 newprinter
hostname: dfA456sourcehost moved to dfA605hostname on newprinter
hostname: cfA456sourcehost converted and moved to cfA605hostname on newprinter
job number is now 605
%
```

Removing print jobs from queue

The `lprm` utility removes jobs from a print queue. Because spooling directories are protected from normal users, `lprm` is the only way a user can remove a job from the queue.

The format of `lprm` is

```
lprm [-Pprinter] [-] [job#] [user]
```

where

`-P`*printer*	specifies the print queue. If the `-P` option is not specified, the default printer queue is used.
-	removes all jobs owned by the user invoking `lprm`. If you are logged in as superuser, this option removes all jobs in the queue.
job#	removes a specific job in the queue identified by its job number.
user	removes all jobs for a user. This option is useful for system managers to remove all files owned by another user.

If `lprm` is used without arguments, it deletes the active job if it is owned by the user who invoked `lprm`.The `lprm` utility displays the names of files it removes. If nothing is displayed, no match was found. When removing files, `lprm` may kill an active daemon. If the daemon is killed, `lprm` automatically restarts a new daemon after removing files from the queue.

An example of an `lprm` request is shown in Figure 22. Note that `lprm` tells when the job has been removed.

Figure 22 Example of `lprm` output

```
% lprm -Pmyprinter 278
hostname: dfA278sourcehost dequeued
hostname: cfA278sourcehost dequeued
%
```

Maintaining striped file systems

5

Various ConvexOS utilities provide means of manipulating and monitoring disks within striped file systems. This chapter discusses tools for maintaining the availability of your file systems, including

- Replacing stripe partitions
- Tracking hot spare status
- Reclaiming hot spare space
- Tape system error messages
- Using VVM to recover from disk failure

Replacing stripe partitions

In some situations, you may wish to replace one disk with another. This section explains how to substitute one disk partition for another in a stripe without dismantling and rebuilding the stripe.

This procedure assumes that all of the disks in question are part of your present disk system. For information on adding disks to your disk system, see the "Setting up the disk system" chapter of *Managing ConvexOS: Configuration Guide.*

For example, suppose partitions du1g and du1h are part of stripes st0 and st1, respectively. st0 is a redundant stripe and is mounted on /foo; st1 is a nonredundant stripe and is mounted on /bar. You want to replace du1g and du1h with du5g and du5h respectively.

Note

It is not necessary to unmount a redundant stripe before using `mvst` on it.

Step 1 Log in as root. If all the partitions to be moved are part of redundant stripes, skip to Step 4.

Step 2 Check any affected nonredundant stripe partitions using `fstat` to make sure there are no open files.

Step 3 For nonredundant stripes, unmount the striped file systems containing the partitions to be replaced using the `umount` command. For example

```
# umount /bar
```

Caution

Double check the arguments you plan to give `mvst` using `mvst -nv` before actually issuing the `mvst` command. If you wish to stop a `mvst`, you must use `kill -15 <pid>` to do so. Stopping `mvst` via any means except a `kill -15` signal results in the loss of all data on the stripe device.

Step 4 Move the stripe data from the old to the new disk partition(s) using the `mvst` command.

```
# mvst st0 du1g du5g dkd-502

# mvst st1 du1h du5h dkd-502
```

Step 5 Remount nonredundant striped file systems using the `mount` command

```
# mount /dev/st1 /bar
```

Tracking hot spare status

When a disk in a redundant stripe fails, the virtual volume manager (VVM) checks to see if a hot spare is available for the affected partitions. If so, VVM reconstructs the data from that disk on the specified hot spare device and continues operations. VVM sends messages to the console regarding the failed disk, notes the percentage of the hot spare used, and keeps track of the amount remaining.

Under normal circumstances, VVM can maintain accessibility of redundant file systems as long as hot spare devices are available. If you choose to use hot spares on your system, track hot spare availability and reclaim space when necessary so that VVM can maintain uninterrupted disk access.

Use the `getst` command as root to view hot spare status. For each hot spare, `getst` displays information in the format shown in Figure 23.

Figure 23 `getst -H` output format

```
# getst -H
hot spare hs0: du9a (64, 2305), sector size 2048 bytes, length 49200 Kbytes
               49200 Kbytes used [100#]
                         affinity towards st0
                         space in use by st0
```

Note that the hot spare in Figure 23 (hs0) is full, and thus would be unavailable for use should a disk in st0 fail.

Reclaiming hot spare space

If the procedure in the previous section reveals that a hot spare is full, or close enough to full that it would be of no use in the event of a crash, use the following procedure to reclaim space on the hot spare:

Step 1 Find out which stripe uses the hot spare.

```
# qst du9a
/dev/rdu9a is used in stripe st1 (redundant)
```

Step 2 For each stripe using the hot spare, select another disk partition to hold the data now residing on the hot spare. The examples in this procedure use du7a as the selected partition.

Step 3 Remove the hot spare to be reclaimed from the hot spare list

```
# rmst -h du9a
```

Step 4 Use the `getst` command to identify the file system mounted on the stripe.

```
# getst st1
stripe st1: sector size 2048 bytes, mounted on /foo
     section a: size 271296 Kbytes/partition,
                      blocking factor 8 Kbytes
     .
     .
```

Step 5 Move the data on the hot spare to the disk you selected in Step 2.

```
# mvst st1 du9a du7a dkd-502
```

Step 6 Remount the file system on the stripe device

```
# mount /dev/st1 /foo
```

Step 7 If more than one stripe uses the hot spare, repeat Step 4 through Step 6 for each stripe.

Step 8 Return the reclaimed hot spare to the hot spare list

```
# newst -H du9a dkd-502
```

Using VVM to recover from disk failure

The remaining sections in this chapter explain messages issued by the Virtual Volume Manager (VMM) and appropriate responses to those messages.

Device failure messages

VVM always sends a message to the console when a device fails. If the state or configuration of your system requires operator-assisted reconstruction, VVM issues additional messages to that effect.

On detection of a failed device, VVM prints the following messages to the console:

```
failure: device /dev/dxxx in stripe /dev/stx
failed
the mvst(8) command to perform the given
reconstruction has begun execution
```

After printing the device failure message to the console, VVM attempts to instruct its daemon, `vvmdaemon`, to begin the automatic reconstruction process. Automatic reconstruction involves using parity or mirrored data to reconstruct data from the failed disk device onto a previously allocated hot spare.

If VVM successfully contacts the daemon, and the daemon is able to perform the reconstruction, no human intervention is required. VVM prints the following message to the console, then places start and finish notifications in the daemonlog:

```
failure: mvst(8) command completed
successfully
the mvst(8) command successfully completed
its task
```

Disk and machine failure

If your machine crashes at the same time a disk fails, the failure indication may not be written to /etc/stripecap. In this case you *must* reconstruct the failed disk data before booting to multiuser mode to prevent losing the data.

Refer to Chapter 10 for the procedure to follow if this situation occurs.

Messages that require operator action

If no hot spare is available, VVM sends the following message to the console:

```
unable to replace device /dev/dxxx in
/dev/stx
this must be done manually with mvst(8)
```

If VVM is unable to contact the daemon, or the daemon is unable to perform the reconstruction, VVM sends the following message to the console:

```
failure: mvst(8) exited with an error
correct the problem and restart the mvst(8)
command manually
```

In the case of any of the above messages, use the `mvst` utility to reconstruct the lost data. Use the procedure described in the following section, "Manual reconstruction".

Manual reconstruction

Follow these steps to reconstruct data from a failed disk.

Step 1 Log in as root.

Because different partitions of the same disk may be included in separate stripes, first determine which stripe(s) included partitions from the failed disk.

Step 2 Enter

```
% qst disk_device
```

disk_device is the name of the failed disk. `qst` returns all stripes that used that disk. For example, suppose `/dev/du4` failed. Then

```
% qst /dev/du4
```

returns

```
/dev/du4h is used in /dev/st1 (non-redundant)
/dev/du4e is used in /dev/st3 (redundant)
```

Perform the following steps for each partition on the failed disk that was part of a redundant stripe. If console messages indicated no suitable hot spare is available, skip to Step 4.

Step 3 Determine whether a suitable hot spare disk is available. Enter

```
% getst -H
```

to view the list of available hot spares. For each hot spare, `getst` returns the amount of space it contains, the stripes (if any) for which it has an affinity, and the locations where the spare partition is used. Figure 24 shows example `getst` output.

Figure 24 Sample `getst` output

```
% getst -H
hot spare hs0: du9a (64, 2305), sector size 2048 bytes, length 49200 Kbytes
          49200 Kbytes used [100%]
                    affinity towards st0
                    space in use by st0
hot spare hs1: du6f (64, 1542), sector size 2048 bytes, length 196800 Kbytes
          49200 Kbytes used [25%]
                    affinity towards st1
                    space in use by st1
```

If a suitable hot spare is available, skip to Step 5.

Step 4 Find an available disk partition (use of the C partition is recommended) that qualifies as a hot spare for the stripe containing the failed disk. To qualify as a hot spare, a partition must have the following qualities:

- Sector size less than or equal to the sector size of the stripe device to be replaced
- Space greater than or equal to that of the partition to be replaced
- Not already used in a stripe with the failed disk
- Same disk type as the failed disk (preferable, but not mandatory)

Step 5 If a hot spare was available, reconstruct data from the failed disk onto the hot spare. Enter

```
% mvst -H /dev/stx /dev/d???
```

where `st`*x* is the stripe containing the failed disk and `d`*???* is the device number of the hot spare partition. If a hot spare was not available, enter

```
% mvst /dev/stx /dev/dbad /dev/d??? xxx-yyy
```

where `st`*x* is the stripe containing the failed disk, `/dev/d`*bad* is the device number of the failed disk, and `/dev/d`*???* and *xxx-yyy* are the device number and device type of the available disk partition that you found in Step 4.

Restarting `vvmdaemon`

Reconstruction is now complete. If hand reconstruction was required because VVM could not contact `vvmdaemon`, you may wish to check for the presence of `vvmdaemon` and restart it if it is not running. To do so, complete the following steps:

Step 1 Enter

```
% ps aux | grep vvmdaemon
```

Step 2 If `vvmdaemon` is not running, enter

```
% vvmdaemon
```

to start it.

Generating accounting reports

6

The ConvexOS accounting system tracks system resources used by individuals and groups of users. From the information collected through the accounting system, you can determine how much disk space, line printer and tape use, or connect time a user or group consumes; how much CPU time is split between users and overhead; and which programs consume the most CPU cycles. With this information, you can

- Plan system use
- Manage system resources
- Keep strict accounting of project costs

This chapter discusses how to generate accounting reports using information collected through the accounting system.

Collecting information

The accounting system is based on users, groups, and activities. Users belong to groups; the tasks they perform are called activities; and a billing account is a group paired with an activity.

The accounting system collects the following types of data:

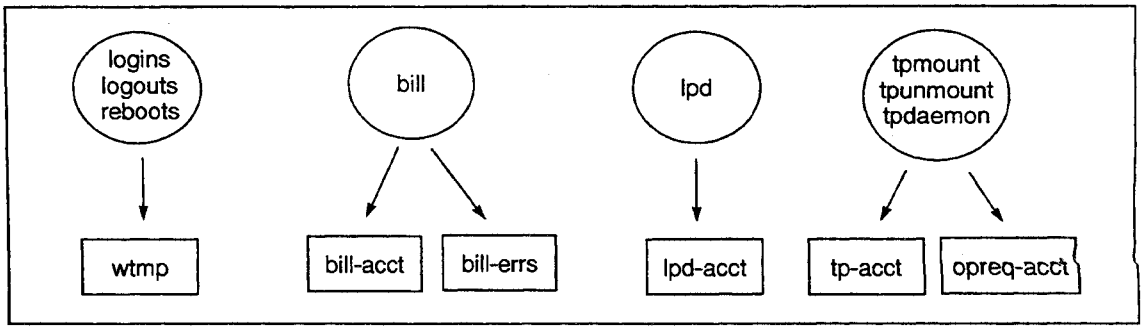
- Process terminations
- Login times
- Successful and unsuccessful executions of `bill`
- Tape allocations and deallocations
- Tape error messages
- Printer use
- Printer use errors

Most accounting data is collected in logs in the /usr/adm directory. This includes

- Process terminations are logged in /usr/adm/acct.
- Login times are collected in /usr/adm/wtmp.
- Successful executions of the `bill` command are collected in /usr/adm/bill-acct.
- Unsuccessful executions of the `bill` command are collected in /usr/adm/bill-errs.
- Tape use data is collected in /usr/adm/tp-acct and /usr/adm/opreq-acct.
- Printer use data is collected in /usr/adm/lpd-acct.

This relationship is illustrated in Figure 25.

Figure 25 Input for accounting log files



Accounting records the following information for each process:

- Process name
- User, group, and activity IDs
- Start time
- Total time run
- Total CPU time
- Total user time
- Total group time
- Controlling tty
- Memory usage
- Disk accesses

Automatic report generation

Daily, weekly, and monthly accounting reports are automatically generated by the `cron` utility. The `cron` utility executes scripts located in /usr/adm named `daily`, `weekly`, and `monthly` at the appropriate times.

The `daily` script copies accounting data from the /usr/adm/acct file to a file named daily#, where # is the day of the week the accounting data is generated. For example, if Sunday is considered the first day of the week, on Sunday accounting data is copied to a file called daily0. Using the `sa` utility, the `daily` script generates the following files from the data stored in the /usr/adm/acct file:

- usracct# (where # is the day of the week the report is generated)
- savacct# (where # is the day of the week the report is generated)
- usracctw
- savacctw
- usracctm
- savacctm
- daily.u

The savacct files contain usage totals for each process, and the usracct files contain usage totals for each user and each unique group/activity combination. Files that end with "w" contain running totals for the week, and files ending with "m" contain running totals for the month. The /usr/adm/daily.u file contains the daily totals for each user in readable text format. After all these files are created, the data in the /usr/adm/acct file is removed.

The `weekly` script generates the following files from the data stored in the usracctw and savacctw files in /usr/adm:

- u.w.*month.day.yr*
- s.w.*month.day.yr*
- week.u

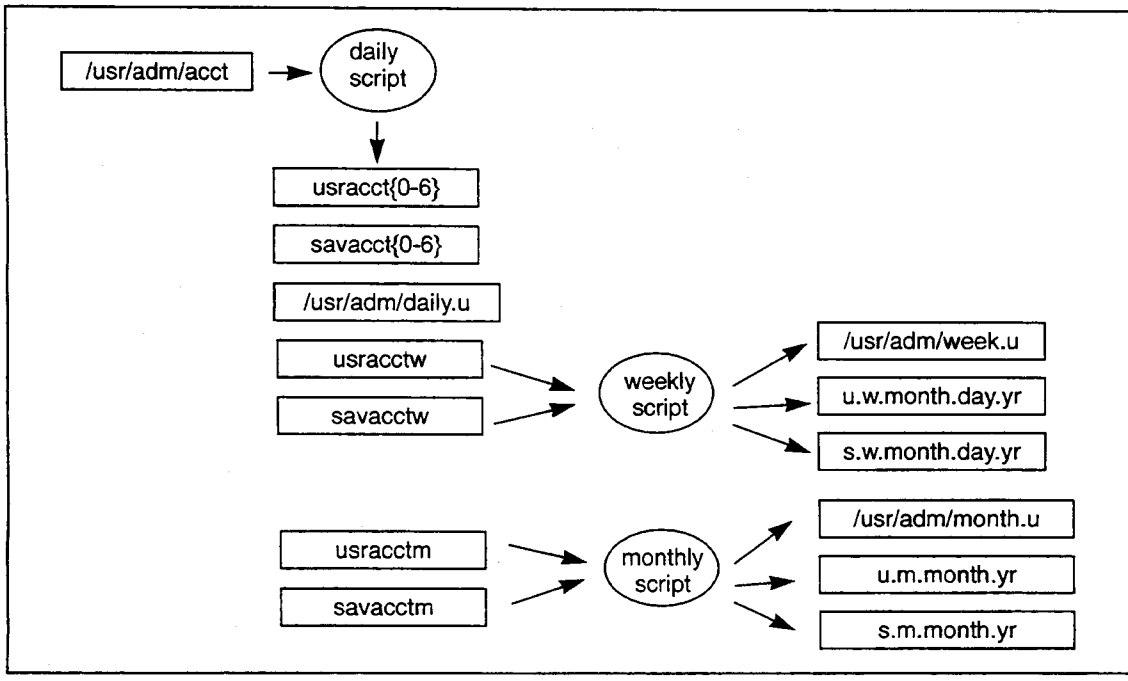
month is the month the report is generated, *day* is the day it is generated, and *year* is the year it is generated. The /usr/adm/week.u file contains weekly totals for each user in text format.

The `monthly` script generates the following files from the data stored in the usracctm and savacctm files:

- u.m.*month.yr*
- s.m.*month.yr*
- month.u

month is the month the report is generated and *year* is the year it is generated. The /usr/adm/month.u file contains monthly totals for each user in text format. Figure 26 illustrates this relationship.

Figure 26 Generated accounting reports



Manual report generation

ConvexOS provides summary utilities to process accounting data. These are

- `connecttime`—The `connecttime` utility processes information from login, logout, and bill executions.
- `pac`—The `pac` utility summarizes printer use data.
- `diskuse`—The `diskuse` utility provides disk use totals for users and groups.
- `sa` and `lastcomm`—The `sa` and `lastcomm` utilities summarize process termination data.

The output from these utilities can be summarized in various formats by piping the output through data-specific `awk` scripts located in `/usr/adm/sumscripts`. These include

- `connecttime.awk`—Pipes the information processed with the `connecttime` utility through the `connecttime.awk` script.
- `sabygrp.awk` and `sabyact.awk`—Pipes the information processed with the `sa` utility through the `sabygrp.awk` and `sabyact.awk` scripts.
  - The `sabygrp.awk` script formats the accounting data by group.
  - The `sabyact.awk` script formats accounting data by activity.
- `diskbygrp.awk`, `diskbyusr.awk`, and `diskmerge.awk`—Pipes the information processed with the `diskuse` utility.
  - The `diskbygrp.awk` script formats the accounting data by group.
  - The `diskbyusr.awk` script formats the accounting data by user.
  - The `diskmerge.awk` script merges totals taken at different times to report running totals over a period of time.

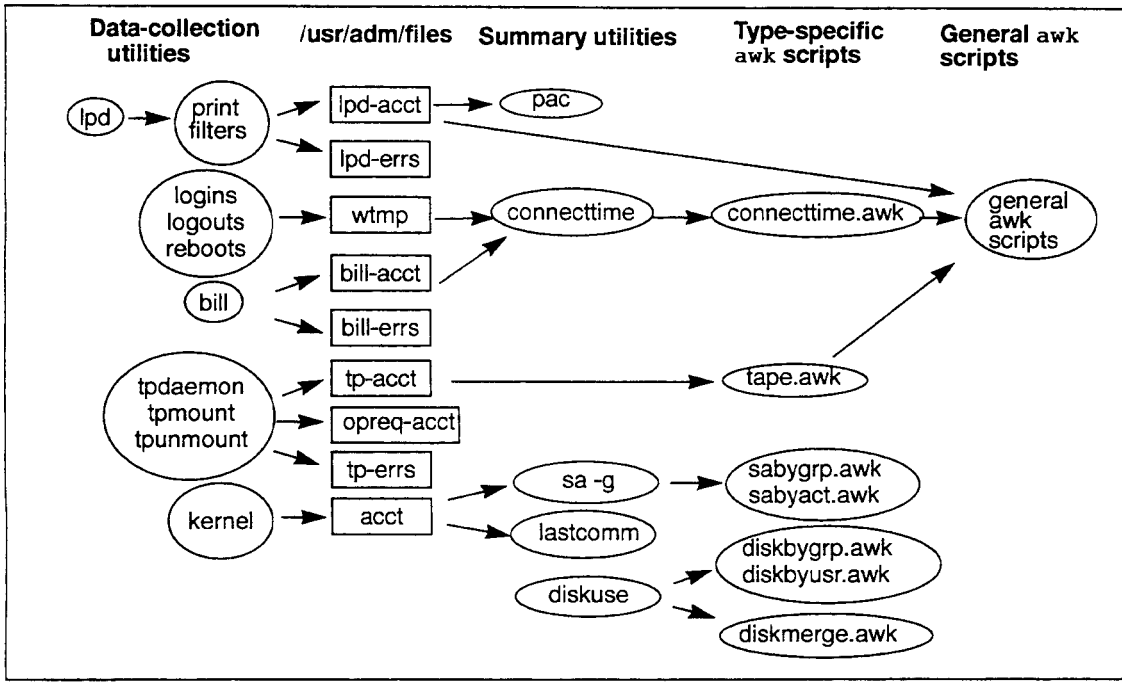
There are also some general summary `awk` scripts located in `/usr/adm/sumscripts` that summarize data in various formats. The general summary `awk` scripts include

- `tape.awk`—Uses the `tape.awk` script to process the information in the `/usr/adm/tp-acct` file.

- `genbyact.awk`, `genbygrp.awk`, `genbygrpact.awk`, and `genbyusr.awk`—Summarizes the `/usr/adm/lpd-acct` output from the `tape.awk` script, and the output from the `connecttime.awk` script.
  - The `genbyact.awk` script summarizes accounting data by activity.
  - The `genbygrp.awk` script summarizes accounting data by group.
  - The `genbygrpact.awk` script summarizes accounting data by group and activity pair.
  - The `genbyusr.awk` script summarizes accounting data by user.

Figure 27 illustrates the relationship between `awk` scripts, accounting files, and summary utilities.

Figure 27 `awk` script relationship to files and summary utilities



In addition, the following two utilities provide standard accounting and old-to-new record conversion options.

The `acctconv` utility converts existing accounting files to a format suitable for use with ConvexOS V11.0. In ConvexOS V11.0. the size of one of the fields of the accounting record has increased, making new accounting records incompatible with records of earlier versions. `acctconv` reads records from a ConvexOS V8.0 or greater accounting file, converts them to V11.0 format, and writes them to a separate file.

The `accton` utility enables or disables standard accounting and periodic accounting. You can configure periodic accounting records to be produced approximately every *n* seconds using the `accton` utility. When a process has been marked to generate a record and at least *n* seconds has elapsed, it produces an accounting record the next time the process makes a system call or the next time the process is scheduled, whichever occurs first.

For more information on these two utilities, refer to the acctconv(8) and accton(8) man pages.

Summarizing process termination data

Process termination data is logged in the /usr/adm/acct file, a binary data file. You can use the `sa` and `lastcomm` utilities to translate the information in the /usr/adm/acct into a text file.

By group and activity combinations

Use the `sa` utility to generate a report on process termination data organized by group and activity combinations. The `sa` command format is

```
sa [options] [acct_file]
```

where

acct_file is the file to process. The default accounting file is /usr/adm/acct.

options control what information is included on the report and the way it is presented. Some of the more common options are listed below. See the sa(8) man page for details on other available options.

`c` Include percentage of total time on report.

`d` Sort records by average number of disk I/O operations.

`D` Print and sort records by average number of disk I/O requests.

`e` Echo records exactly as they appear in the accounting file.

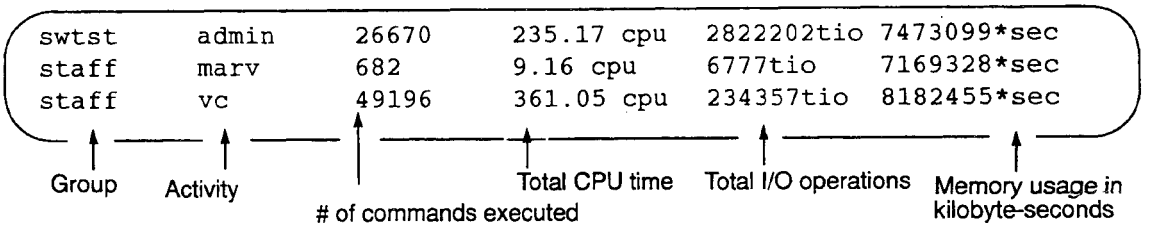
`g` Print totals for each group and activity combination.

You can direct the output to a file; otherwise, it is printed on the display screen. For example, to generate a file called report that contains accounting information for each group and activity combination, enter the following command:

```
% sa -g > report
```

Figure 28 illustrates the output you would receive from this command.

Figure 28 Sample `sa -g` output



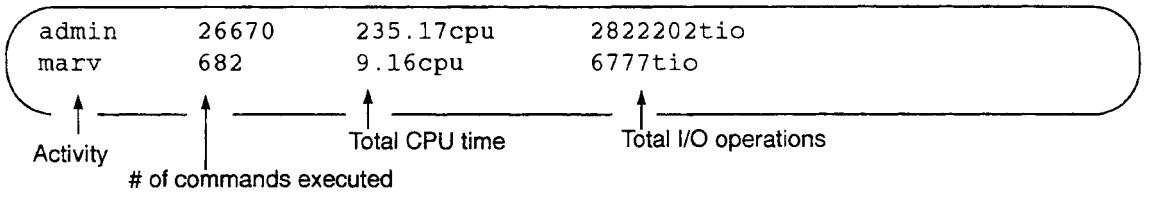
You can optionally summarize the output of the `sa -g` command by piping it through one of two sa awk scripts located in /usr/adm/sumscripts: `sabyact.awk` or `sabygrp.awk`.

Use the following command to pipe `sa -g` output through the `sabygrp.awk` script:

```
% sa -g | awk -f /usr/adm/sumscripts/sabyact.awk
```

The `sabyact.awk` script summarizes the accounting data by activity. Figure 29 illustrates the output from the `sabyact.awk` script.

Figure 29 Sample `sabyact.awk` script output

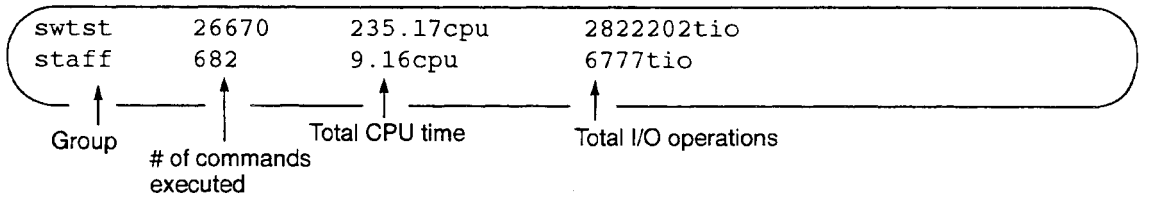


Use the following command to pipe `sa -g` output through the `sabygrp.awk` script:

```
% sa -g | awk -f /usr/adm/sumscripts/sabygrp.awk
```

The `sabygrp.awk` script summarizes the accounting data by group. Figure 30 illustrates the output from the `sabygrp.awk` script.

Figure 30 Sample `sabygrp.awk` output



By command execution sequence

Use the `lastcomm` utility to generate a report on process termination data organized in order of most recent to least recent command executed. The command format is

`lastcomm [-f` *acct_file*`] [`*command*`] [`*user_name*`] [`*term_name*`]`

which has the following components:

acct_file	File to process. The default file is /usr/adm/acct. You can specify an alternate file name using the `-f` option.
command	Command to match
user_name	User name to match
term_name	Terminal name to match

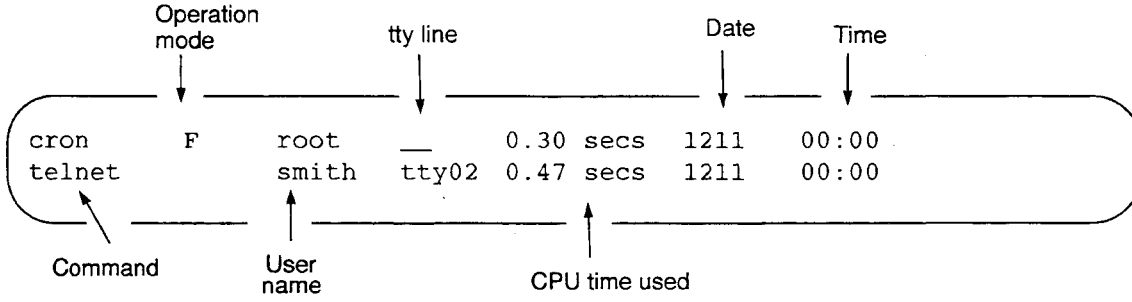
When executed with no arguments, information about all commands recorded in the specified accounting file is printed. If arguments are included, only the accounting entries that match the argument criteria are used. For example, the following command produces a listing of all executions of commands named `a.out` by user root on terminal ttyd0:

```
% lastcomm a.out root ttyd0
```

See the lastcomm(8) man page for more details on using this utility.

Figure 31 illustrates sample output from this command.

Figure 31 Sample `lastcomm` utility output



The output of Figure 31 contains the following fields from left to right

- Command name of the process
- Modes of operation. This could be any of the following values:

  `S` Command executed with superuser privileges

  `F` Command ran after a fork without a following exec

  `D` Command terminated with generation of a core file

  `X` Command terminated with a signal

  `f` Command ran as a fixed-schedule job at least part of the time
- User name
- tty line
- CPU time used

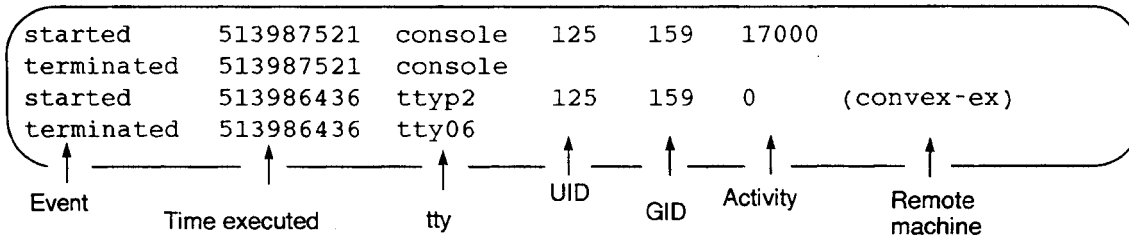
Summarizing login and logout data

Login, logout, and reboot data are logged in the /usr/adm/wtmp file, and bill execution data is logged in the /usr/adm/bill-acct file. Both files are binary files. The `connecttime` utility translates the information in these binary files into an ASCII text file that contains when, on which device, and by whom logins, logouts, executions, and termination of bill occur. The `connecttime` command format is

```
connecttime
```

Figure 32 illustrates sample output from this command.

Figure 32 Sample `connecttime` utility output



The most recent entry appears first. The output contains the following fields from left to right:

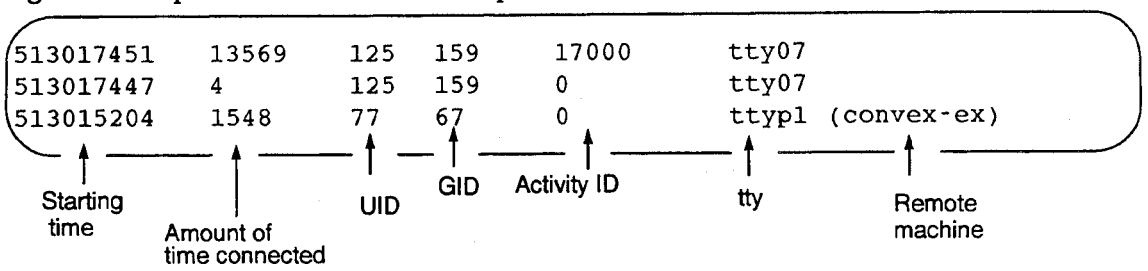
- Whether the connection started or terminated
- Starting or ending time in seconds
- tty line
- User ID
- Group ID
- Activity ID
- Remote machine from which the user logged in, if login is from remote machine.

You can use the `connectime.awk` script to summarize this data. For example, enter the following command to pipe the output from the `connecttime` utility through the `connecttime.awk` script:

```
% connecttime | awk -f /usr/adm/sumscripts/connecttime.awk
```

Figure 33 illustrates sample output from this command.

Figure 33 Sample `connecttime.awk` output



The output contains the following fields, from left to right:

- Starting time measured in seconds elapsed since 00:00:00 GMT, January 1, 1970
- Time connected in seconds
- User ID (-1 if user is not in /etc/passwd file)
- Group ID
- Activity ID
- tty line
- Remote machine from which the user logged in

You can optionally run the output of the `connecttime.awk` script through the general summary `awk` scripts:

- The `genbyact.awk` script summarizes accounting data by activity.
- The `genbygrp.awk` script summarizes accounting data by group.
- The `genbygrpact.awk` script summarizes accounting data by group and activity pair.
- The `genbyusr.awk` script summarizes accounting data by user.

For example, to pipe the output of the `connecttime.awk` script through the `genbygrpact.awk` script and send it to a file called connect.rep in /tmp, enter

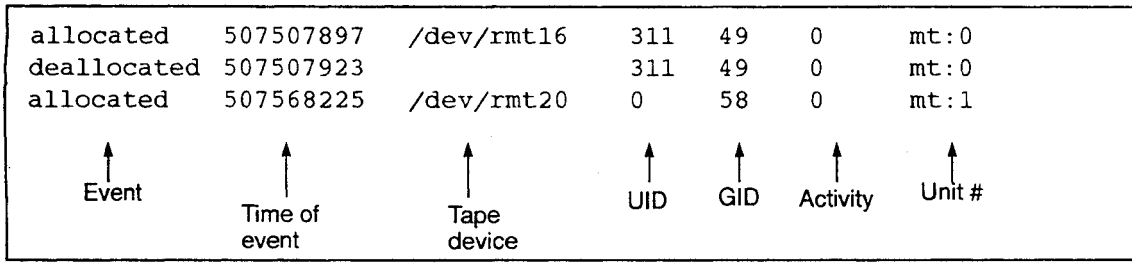
```
% connecttime | awk -f /usr/adm/sumscripts/connecttime.awk |          \
awk -f /usr/adm/sumscripts/genbygrpact.awk > /tmp/connect.rep
```

If you do not specify an output file, results are displayed on the terminal screen.

Summarizing tape use data

Tape use data is logged in the /usr/adm/tp-acct file. This file is a text file and can be viewed using the `cat`, `less`, or `more` utilities. Figure 34 illustrates a sample tp-acct file.

Figure 34 Sample /usr/adm/tp-acct file



This file contains the following fields, from left to right:

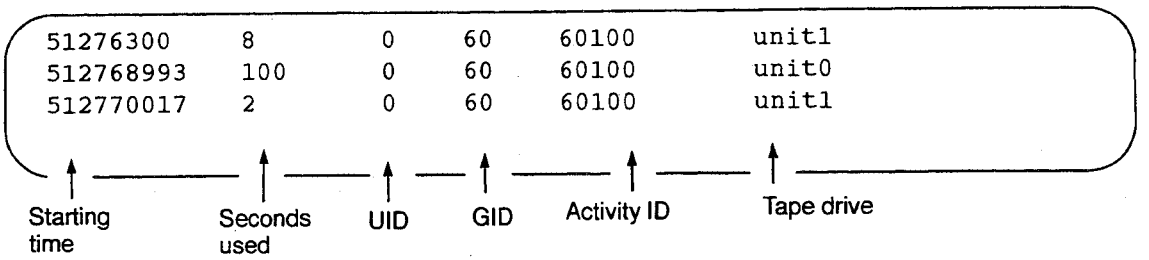
- Whether tape drive was allocated or deallocated
- Time tape drive was mounted or unmounted
- Tape device
- User ID
- Group ID
- Activity ID
- Tape device unit number

You can optionally run the data in the /usr/adm/tp-acct file through the `tape.awk` script. This script converts data in the /usr/adm/tp-acct file to a format that can be processed by the general summary `awk` scripts. Enter the following command to do this:

```
% awk -f /usr/adm/sumscripts/tape.awk /usr/adm/tp-acct > tape.rep
```

If you do not enter an output file name, results display on the terminal screen. The output in this example is saved to the file tape.rep. Figure 35 illustrates the output for the `tape.awk` command.

Figure 35 Sample `tape.awk` output



The output contains the following fields, from left to right:

- Starting time, measured in seconds elapsed since 00:00:00 GMT, January 1, 1970
- Amount of time in seconds tape drive was allocated
- User ID
- Group ID
- Activity ID
- Logical name of tape drive

You can optionally pipe the output of the `tape.awk` script through the general summary `awk` scripts:

- The `genbyact.awk` script summarizes accounting data by activity.
- The `genbygrp.awk` script summarizes accounting data by group.
- The `genbygrpact.awk` script summarizes accounting data by group and activity pair.
- The `genbyusr.awk` script summarizes accounting data by user.

For example, to pipe the output of the `tape.awk` script through the `genbygrpact.awk` script and send it to a file called tape.rep in /tmp, enter the following command:

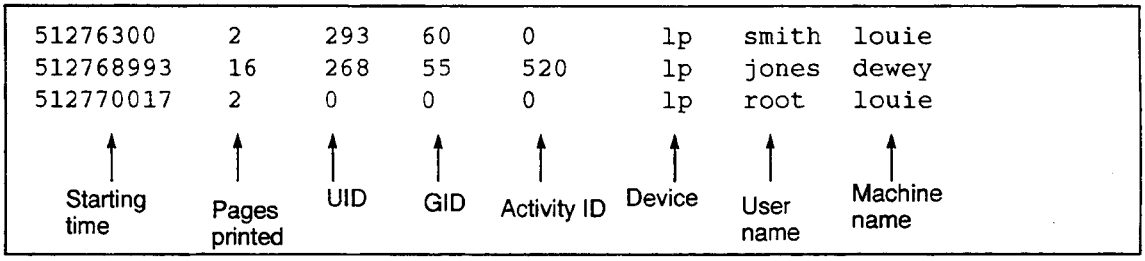
```
% awk -f /usr/adm/sumscripts/tape.awk /usr/adm/tp-acct | awk -f \
/usr/adm/sumscripts/genbygrpact.awk > /tmp/tape.rep
```

If you do not specify an output file, the results are displayed on the terminal screen.

Summarizing printer use data

Printer use data is logged in the /usr/adm/lpd-acct file. This file is an ASCII file and can be viewed using the `cat`, `less`, or `more` utilities. Figure 36 illustrates a sample lpd-acct file.

Figure 36 Sample lpd_acct file



This file contains the following fields, from left to right:

- Time printer was used
- Number of pages printed
- User ID
- Group ID
- Activity ID
- Device used
- User name
- Machine name

You can optionally process the printer accounting data in the /usr/adm/lpd-acct file with the `pac` summarizing utility or the general summary awk scripts.

Using the pac summarizing utility

Enter the following command to process the printer accounting data using the pac utility:

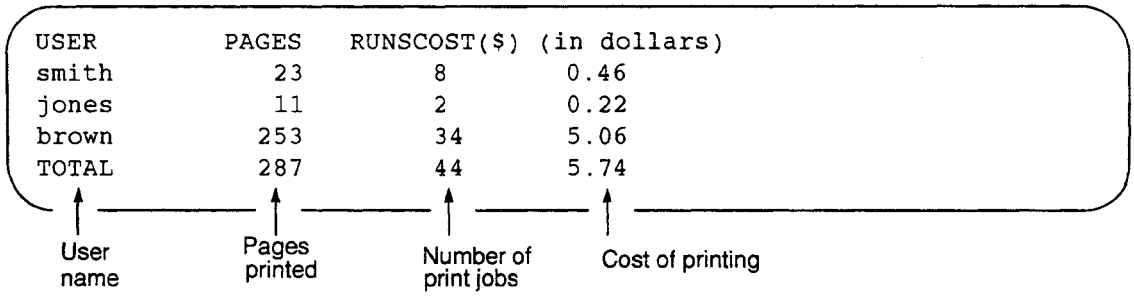
```
% pac
```

The output is sorted alphabetically by user, but this can be controlled by specifying different options. Some of the more common options are listed below. See the pac(8) man page for details on other available options.

- `c` Sort the output by cost.
- `r` Reverse the order of the output.
- `s` Place summary data in /usr/adm/lpd-acct_sum.

Figure 37 shows sample pac output.

Figure 37 Sample pac utility output



This output contains the following fields, from left to right

- User name
- Number of pages printed
- Number of print jobs executed
- Total cost for printing

The cost is arbitrary. See the pac(8) man page for the details on how to modify the cost equation.

Using general summary scripts

You can optionally process the data in the /usr/adm/lpd-acct file through the general summary awk scripts. This is the recommended method.

- The `genbyact.awk` script summarizes accounting data by activity.
- The `genbygrp.awk` script summarizes accounting data by group.
- The `genbygrpact.awk` script summarizes accounting data by group and activity pair.
- The `genbyusr.awk` script summarizes accounting data by user.

For example, to process printer use data through the `genbygrp.awk` script and send it to a file called printer.rep in /tmp, enter

```
% awk -f /usr/adm/sumscripts/genbygrp.awk /usr/adm/lpd-acct > \
/tmp/printer.rep
```

If you do not specify an output file, the results are displayed on the terminal screen.

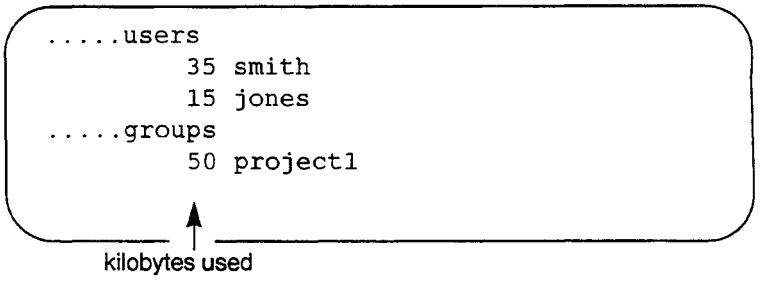
Summarizing disk use data

Disk use accounting data is run through a combination of standard utilities, shell scripts, and awk scripts. The main disk use utility is `diskuse`. This utility displays disk usage in kbytes for a directory by user and group. To override directory protections, log in as superuser to run `diskuse`. The `diskuse` command format is shown below:

`diskuse [-d` *directory*`]`

The top of the directory hierarchy is the default. You can specify an alternate directory using the `-d` argument. Figure 38 illustrates sample `diskuse` utility output.

Figure 38 Sample `diskuse` utility output



The `diskmerge.awk` script merges two versions of `diskuse` output to create totals for each user and group. You can use the `diskmerge.awk` script to keep track of disk use over a period of time by adding current usage data to old data. For example, yesterday you stored `diskuse` totals in a file called diskuse.out. Today, you want to merge `diskuse` totals with yesterday's to get a two-day total. To do this, enter

```
% diskuse | cat - diskuse.out | awk -f\
/usr/adm/sumscripts/diskmerge.awk
```

Output for this command is the same as that shown in Figure 38.

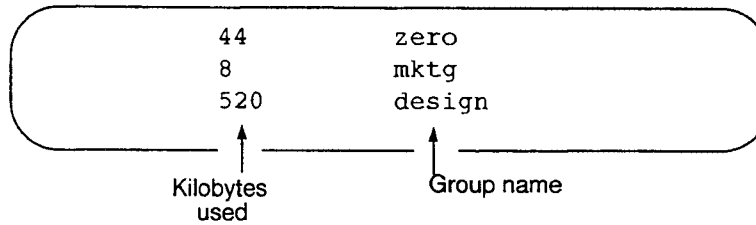
Output from the `diskuse` utility is in a form suitable for processing through `diskbygrp.awk` and `diskbyusr.awk` scripts.

The `diskbygrp.awk` script summarizes disk use by group. To summarize `diskuse` data from the /mnt directory by group, enter

```
% diskuse -d /mnt | awk -f /usr/adm/sumscripts/diskbygrp.awk > \
diskuse.rep
```

If you do not specify an output file, results are displayed on the terminal screen. Figure 39 illustrates the output from this command.

Figure 39 Sample `diskbygrp.awk` output



This output contains the following fields, from left to right:

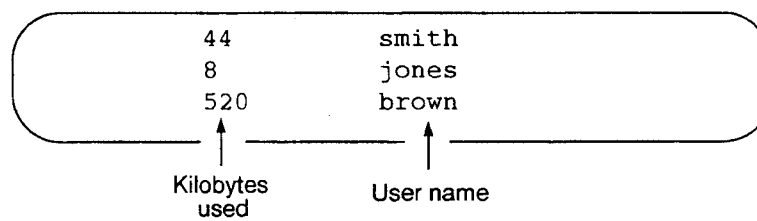
- Total disk space used in kilobytes
- Group name

The `diskbyusr.awk` script summarizes disk use by users. To summarize `diskuse` data from the /mnt directory by user, enter

```
% diskuse -d /mnt | awk -f /usr/adm/sumscripts/diskbyusr.awk > \
diskuse.rep
```

If you do not specify an output file, results are displayed on the terminal screen. Figure 40 illustrates the output from this command.

Figure 40 Sample `diskbyusr.awk` output



This output contains the following fields, from left to right:

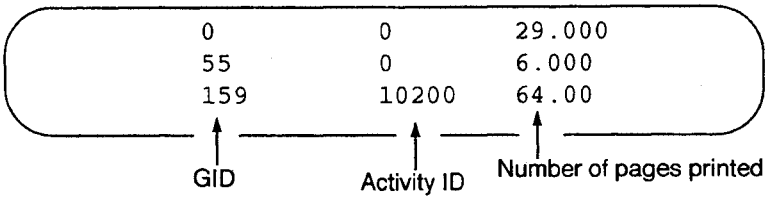
- Total disk space used in kilobytes
- User name

Miscellaneous utilities

The `sort` and `idtoname` utilities are helpful for summarizing accounting information. You can pipe output from general summary scripts through `sort`, then through `idtoname` to get summarized, sorted, and readable results. The `idtoname` utility converts user, group, and activity IDs, as well as any time stamps to user name, group name, activity name, and date respectively.

For example, Figure 41 illustrates a sample of the output you would receive if you run the /usr/adm/lpd-acct file through the general `genbygrpact.awk` script.

Figure 41 Sample `genbygrpact.awk` output

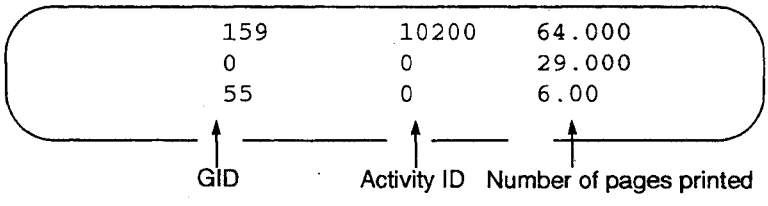


To display output in order of most number of pages printed, pipe the output through `sort`. For example:

```
% awk -f /usr/adm/sumscripts/genbygrpact.awk /usr/adm/lpd-acct |      \
sort +2nr
```

The number 2 indicates the number of fields to skip, `n` is the type of data in the field being sorted (numeric), and `r` specifies sorting in descending order. This command sorts output from `genbygrpact.awk` in reverse order based on the third field in each line. Figure 42 illustrates the output you would get from this command. Compare this to the output you received from the same file in Figure 41.

Figure 42 Sample sorted `genbygrpact.awk` output

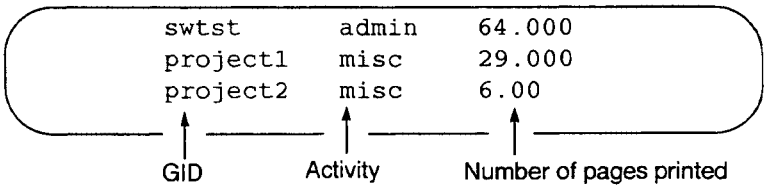


To display output by group and activity name rather than user and activity IDs, pipe the output through the `idtoname`. For example:

```
% awk -f /usr/adm/sumscripts/genbygrpact.awk /usr/adm/lpd-acct          \
| /usr/bin/sort +2nr | /usr/convex/idtoname -gae
```

The `g` option specifies that the first field is a group ID, the `a` option specifies the second field is an activity ID, and the `e` option specifies to echo the third field as it exists. Figure 43 illustrates the output you would get from this command. Compare this to the output you received from the same file in Figure 41.

Figure 43 Sample sorted and `idtoname genbygrpact.awk` output



Managing sendmail 7

ConvexOS V11.0 includes V8.6 of sendmail, the internetwork mail routing program. This chapter provides information for managing sendmail. Major tasks covered are

- Where to find sendmail documentation
- Supported products
- Starting and stopping sendmail
- Running the sendmail daemon
- Printing the queue
- Forcing the queue
- Load limiting

Where to find sendmail documentation

For complete sendmail information, CONVEX recommends that you refer to the O'Reilly & Associates sendmail book titled *"sendmail,"* written by Brian Costales, with Eric Allman and Neil Rickert.

Refer to *Managing ConvexOS: Configuration Guide,* Chapter 10 for

- General information on the changes to sendmail with the V11.0 release of ConvexOS.
- Differences between the O'Reilly documentation and ConvexOS's sendmail.

Supported products

sendmail provides (invisibly to users) support for the CONVEX Fair Share Scheduler and ConvexOS/Secure.

Starting and stopping `sendmail`

`sendmail` is usually started in the /etc/rc.std file when the system reboots. There are also times, such as when restarting the `sendmail` daemon, that you start `sendmail` from the command line.

This section explains how to run the `sendmail` daemon and freeze the configuration file for quicker `sendmail` start up. Refer to the sendmail(8) man page for a complete listing of `sendmail` operating modes and options.

Running the `sendmail` daemon

`sendmail` is run in daemon mode by specifying the `-bd` flag. The `sendmail` daemon must be running to

- Receive incoming mail over an IPC connection
- Process the mail queue

The `sendmail` daemon listens for SMTP connection requests on port 25 and forks SMTP servers to accept the connections. When run in this mode, `sendmail` alters its name as seen by `ps` to show what task it is performing. The most common names are

- `accepting connections`
- `rejecting connections: load average` *<average>*
- `running queue`
- `srvrsmtp` *<hostname>*

When `sendmail` is run with the *-qinterval* flag, the queue processing daemon wakes up at the specified interval to process the mail queue. Valid values for the time interval are shown in Table 1.

Table 1 Time interval values

Value	Meaning
s	Seconds
m	Minutes
h	Hours
d	Days
w	Weeks

Printing the queue

You can print the contents of the mail queue with the command

```
% /usr/lib/sendmail -bp
```

or

```
% mailq
```

This produces a listing of the queue IDs, the message size, the date the message entered the queue, and the sender and recipients. If you run `sendmail` or `mailq` with the `-v` flag, the output also includes the message priority. Refer to the mqueue(5) man page for information on queue file contents and format.

Figure 44 shows an example of output of the `mailq` command.

Figure 44 Mail queue contents

```
Mail Queue (2 requests)
--QID----Size-------Q-Time----------Sender/Recipient---------
AA13731    5152     Tue Aug 6 16:15             jsmith
                                                mjones
AA13737    9092     Tue Aug 6 16:16             peters
                                                dlyons
```

Forcing the queue

`sendmail` is usually started in the /etc/rc.std file with the option to process the queue, or redeliver queued messages, at the specified time interval. (Refer to the section, "Running the sendmail daemon," on page 84.) However, if the queue becomes clogged, you can force `sendmail` to process the queue immediately by invoking `sendmail` with the `-q` flag and no specified interval.

Load limiting

There are two `sendmail` configuration options you can use to limit load averages on your system: the `x` and `X` options.

If the load average on your system gets too high, you can use the `x` option to have `sendmail` queue low priority messages rather than deliver them. When the five-minute load average exceeds the value of the `x` option, `sendmail` queues messages if the queue factor (set with the `q` option), divided by the difference in the current load average and the load average limit plus one, exceeds the priority of the message.

That is, if the load average is greater than the load average limit (`x`), the message is queued if the priority value is greater than

```
queue factor/(load average - load average limit + 1)
```

This means that when the load average reaches the load average limit, the `q` option defines the minimum priority value (maximum priority) of messages that are queued. The default value of the `q` option is 10000. The value of the `x` option is set to 8 in the supplied configuration file.

The `X` option defines a load average at which `sendmail` refuses to accept network connections. Locally generated mail, including incoming UUCP mail, is still accepted.

Table 2 lists recommended values for the x and X options for various CONVEX systems. These values depend on site configuration, the applications that are running, and the importance of prompt mail delivery at your site.

Table 2 Recommended x and X option values

Hardware	Mail is low priority		Mail is high priority (deliver on a heavily loaded schedule)	
C1	x=4	X=8	x=8	X=12
C210/C3210	x=6	X=10	x=10	X=14
C220/C3220	x=8	X=12	x=12	X=16
C230/C3230	x=10	X=14	x=14	X=18
C240/C3240	x=12	X=16	x=16	X=20
C3410	x=6	X=10	x=10	X=14
C3420	x=8	X=12	x=12	X=16
C3430	x=10	X=14	x=14	X=18
C3440	x=11	X=16	x=16	X=20
C3810	x=10	X=18	x=16	X=22
C3820	x=12	X=20	x=20	X=24
C3830	x=14	X=22	x=22	X=26
C3840	x=16	X=24	x=22	X=28
C3850	x=18	X=26	x=24	X=30
C3860	x=20	X=28	x=26	X=32
C3870	x=22	X=30	x=28	X=34
C3880	x=24	X=32	x=30	X=38

Checking the file system

8

Maintaining file system integrity is one of the most critical functions of a system manager. File systems can easily become corrupted if inconsistencies are not caught and corrected in time. This chapter presents an overview of the ConvexOS file system and describes how to use the file system check (`fsck`) utility to find and correct file system problems.

Internal view of the file system

The ConvexOS file system controls access to data files as well as devices on the system. It provides a hierarchical organization for files and directories, buffering of data, and control of file access. Before describing how `fsck` repairs the file system, knowing the internal organization of the ConvexOS file system is useful. The following sections give a brief overview of relevant information.

For information about creating file systems, refer to Chapter 3, "Setting up the disk system," in the *Configuration Guide*.

The superblock

A file system is described by its *superblock*. The superblock is built when the file system is created (with `newfs` or `newst`) and never changes. It contains the basic parameters of the file system, such as the number of data blocks in the file system and a count of the maximum number of files. Because the superblock contains critical data, `newfs` replicates it across the disk partition to protect against catastrophic loss. The default superblock always resides at a fixed offset from the beginning of the partition. Redundant superblocks are not used unless a head crash or some other disk error corrupts the default superblock.

Summary information

Nonreplicated summary information is associated with the superblock. The summary information, which changes as the file system is modified, contains the number of blocks, fragments, inodes, and directories in the file system.

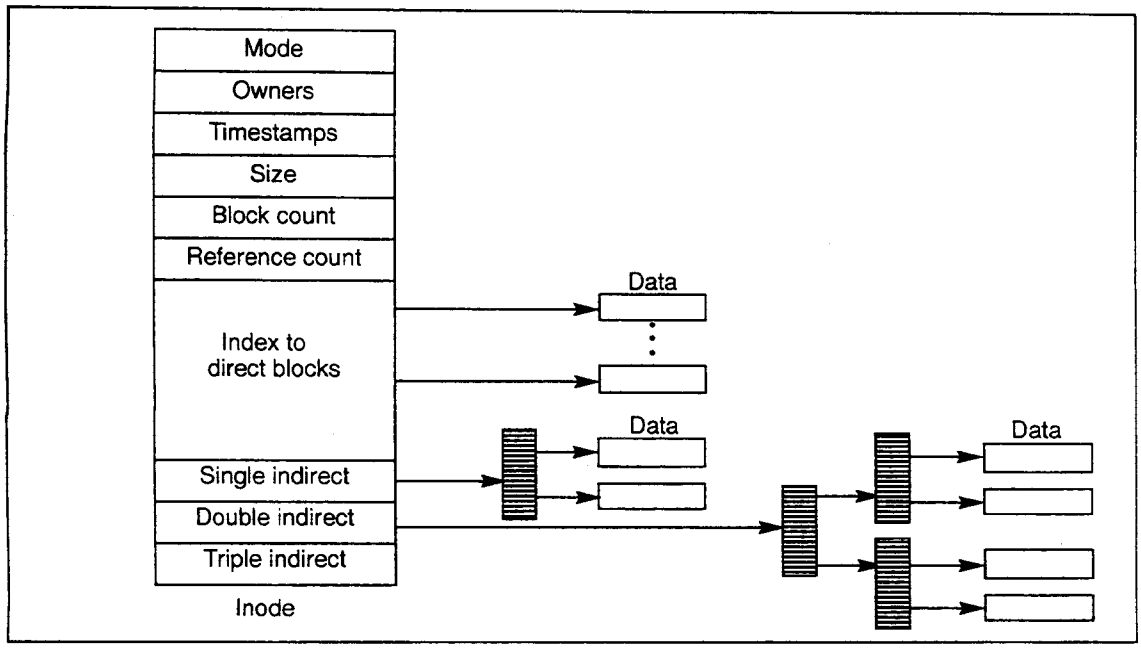
Data blocks

A file system is divided into units called data blocks or, simply, blocks. A block is a contiguous sequence of data. File system data blocks can be 4K, 8K, 16K, 32K, or 64K bytes in size; each character in a file is represented by a byte of information. A file may reside in many blocks located on different portions of the disk. Ideally, blocks that form a file should be contiguous to optimize disk performance. However, as the disk is used and files are deleted, free blocks become increasingly scattered. To handle this, the file system uses inodes to point to noncontiguous blocks associated with a file.

Inodes

Every file in the file system has an associated descriptor called an index node, or *inode*. An inode contains information describing ownership of the file, time stamps indicating modification and access times for the file, and an array of indexes pointing to the data blocks for the file. Figure 45 shows the structure of an inode.

Figure 45 Inode structure



The first 12 blocks of a file are directly referenced by values stored in the inode structure itself. Additional data blocks are described with a pointer from the inode to an indirect data block as shown in Figure 45. When accessing a file over 12 blocks long, the system must first fetch the indirect block, which stores the number of the data block.

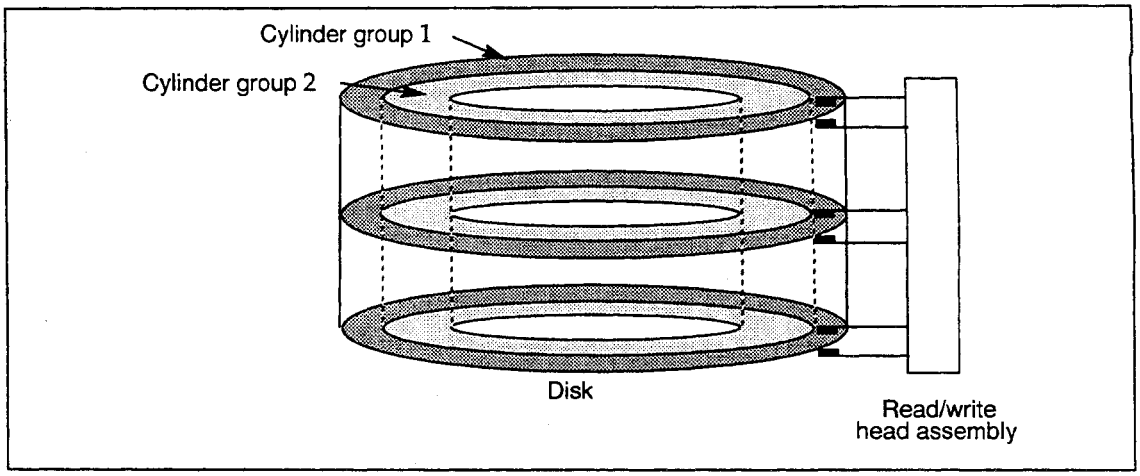
An indirect block that contains data block numbers is called a single indirect block; an indirect block that contains block numbers of single indirect blocks is called a double indirect block. In a file system with a 4096-byte block size, a single indirect block contains 1024 more block addresses, a double indirect block contains 1024 addresses of single indirect blocks, and a triple indirect block contains 1024 addresses of double indirect blocks.

To create files up to 2^{31} bytes using only two levels of indirection, the minimum size of a file system block is 4096 bytes. The size of file system blocks can be any power of two greater than or equal to 4096. The block size of the file system is maintained in the superblock, so it is possible for file systems of different block sizes to be accessible simultaneously on the same system. The block size must be decided when `newfs` creates the file system; it cannot be changed subsequently without rebuilding the file system.

Cylinder groups

The file system partitions a disk into one or more areas called cylinder groups. A cylinder group is made up of one or more consecutive cylinders on a disk. Figure 46 illustrates the layout of cylinder groups on a disk in the ConvexOS file system.

Figure 46 Layout of cylinder groups



Each cylinder group includes inode slots for files, a block map describing available blocks in the cylinder group, and summary information describing usage of data blocks within the cylinder group. A fixed number of inodes is allocated for each cylinder group when the file system is created. Typically, one inode is allocated for each 2048 bytes of disk space; this is usually many more inodes than is ever needed.

Fragments

To avoid file system waste when storing small files, the file system space allocator divides a single file system block into one or more fragments. The fragmentation of the file system is specified when the file system is created; each file system block can be broken into 1, 2, 4, or 8 addressable fragments. The smallest fragment size is constrained by the disk sector size which is, typically, 512 bytes (2 kbytes for IDC disks). The block map associated with each cylinder group records the space availability at the fragment level. Aligned fragments are examined to determine block availability.

On a file system with a block size of 4096 bytes and a fragment size of 1024 bytes, a file is represented by zero or more 4096-byte blocks of data, and possibly a single fragmented block. If a block must be fragmented to obtain space for a small amount of data, the remainder of the block is made available to other files. For example, consider an 11,000-byte file stored on a file system with a 4096-byte block size and a 1024-byte fragment size. This file uses two full-size blocks and a 3072-byte fragment. If no fragments with at least 3072 bytes are available when the file is created, a full-size block is split providing the 3072-byte fragment and an unused 1024-byte fragment. This remaining fragment can be allocated to another file as needed.

How file system inconsistencies occur

As you create and modify files, the file system tracks inode allocation, the number of blocks in each file, and the total number of free blocks and inodes. This information is maintained in memory until the disks are updated by the `sync` or `update` commands, or the `sync` system call. Unfortunately, not all information regarding a file is written to disk at the same time. For example, the system may update the inode and write the data blocks for a given file to disk, but not update the superblock until later. If the system crashes or is corrupted during this time, the file system will not be consistent because information left in memory was not written to disk before the crash occurred.

Caution

A crash is not the only cause of file system corruption. Hardware failures, improper shutdown procedures, or operator errors such as taking a disk offline without unmounting file systems or write-protecting a mounted file system can also cause file systems to become corrupted. If inconsistencies are found but not corrected, the file system may become so severely corrupted that it is not repairable.

File system information checked

Inconsistencies in the file system can occur in a number of areas. The following sections describe what can go wrong in a file system and how `fsck` finds inconsistencies.

Note

The `fsck` utility is usually run noninteractively. In this mode, it only fixes corruptions that you would normally expect from an improper system shutdown. However, for the remainder of this chapter, `fsck` is described from the interactive standpoint.

Checking the superblock

The most commonly corrupted item in a file system is the summary information associated with the superblock. Summary information is prone to corruption because it is modified every time data blocks or inodes are modified.

The superblock is checked for inconsistencies in file system size, number of inodes, free block count, and free inode count. File-system size must be larger than the number of blocks used by the superblock and the number of blocks used by the list of inodes. File-system size and layout information are the most critical pieces of information that `fsck` checks. While there is no way to check these sizes (because they are statically determined by `newfs`), `fsck` can verify that sizes are within reasonable bounds. All other file system checks require that sizes be correct. If `fsck` detects corruption in the static parameters of the default superblock, it requests the location of an alternate superblock.

Free block checking

The `fsck` utility checks that all blocks marked as free in the cylinder-group block maps are not claimed by any files. When all the blocks have been initially accounted for, `fsck` checks that the number of free blocks plus the number of blocks claimed by the inodes equals the total number of blocks in the file system.

If anything is wrong with the block allocation maps, `fsck` rebuilds them, based on the list of allocated blocks it has calculated.

Summary information associated with the superblock counts the total number of free blocks within the file system. `fsck` compares this count to the number of free blocks it found. If the two counts do not agree, `fsck` replaces the incorrect count in the summary information with the actual free-block count.

The summary information also counts the total number of free inodes within the file system. `fsck` compares this count to the number of free inodes it found. If the two counts do not agree, `fsck` replaces the incorrect count in the summary information with the actual free inode count.

Checking the inode state

An individual inode is not as likely to be corrupted as the allocation information. However, because of the great number of active inodes, a few inodes are usually corrupted.

The `fsck` utility sequentially checks the list of inodes in the file system, starting with inode 2 (erased files have their inode number set to 0; inode 1 is reserved for future use). `fsck` progresses through the list until it reaches the last inode in the file system. It checks the state of each inode for inconsistencies in format and type, link count, duplicate blocks, bad blocks, and inode size.

Each inode contains a mode word. This mode word describes the type and allocation state of the inode. Inodes must be one of seven types:

- Regular
- Directory
- Symbolic link
- Block special
- Character special
- Socket
- Named pipe (fifo)

Inodes may have one of three allocation states:

- Unallocated
- Allocated
- Neither unallocated nor allocated

The last state suggests an incorrectly formatted inode. This can occur if bad data is written into the inode list. The only possible corrective action is for `fsck` to clear the inode.

Checking inode links

Each inode contains a count of the total number of directory entries linked to the inode. The `fsck` utility verifies the link count of each inode by starting at the root of the file system, and descending through the directory structure. The actual link count for each inode is calculated during the descent.

If the stored link count is nonzero and the actual link count is zero, there is no directory entry for the inode; `fsck` places the disconnected file in the lost+found directory. If the stored and actual link counts are nonzero and unequal, a directory entry may have been added or removed without the inode being updated. If this happens, `fsck` replaces the incorrect stored link count with the actual link count.

Each inode contains a list, or pointers to lists (indirect blocks), of all the blocks claimed by the inode. Because indirect blocks are owned by an inode, inconsistencies in an indirect block directly affects the inode that owns it.

The `fsck` utility compares each block number claimed by an inode against a list of already allocated blocks. If another inode already claims a block number, the block number is added to a list of duplicate blocks. Otherwise, the list of allocated blocks is updated to include the block number.

If there are duplicate blocks, `fsck` performs a partial second pass over the inode list to find the inode of the duplicated block. The second pass is needed: without examining the files associated with these inodes for correct content, not enough information is available to determine which inode is corrupted and should be cleared. If this condition does arise, the inode with the earliest modification time is usually incorrect, and should be cleared. However, `fsck` prompts you to clear both inodes; you must decide which one to keep and which one to clear.

`fsck` checks the range of each block number claimed by an inode. If the block number is lower than the first data block in the file system, or greater than the last data block, the block number is a bad block number. When an inode has many bad blocks, it is usually caused by an indirect block that was not written to the file system. If an inode contains bad block numbers, `fsck` prompts you to clear it.

Checking inode data size

Each inode contains a count of data blocks it contains. The number of actual data blocks is the sum of allocated data blocks and indirect blocks. `fsck` computes the actual number of data blocks and compares that block count against the number of actual blocks the inode claims. If an inode contains an incorrect count, `fsck` prompts you to fix it.

Each inode contains a 64-bit size field. Size is the number of data bytes in the file associated with the inode. The consistency of the byte size field is roughly checked by computing from the size field the maximum number of blocks that should be associated with the inode, and comparing that expected block count against the actual number of blocks the inode claims.

Checking data associated with an inode

An inode can directly or indirectly reference three kinds of data blocks:

- Plain data blocks contain the information stored in a file.
- Symbolic link data blocks contain the path name stored in a link.
- Directory data blocks contain directory entries. `fsck` can check only the validity of directory data blocks.

All referenced blocks must be the same kind.

Each directory data block is checked for the following inconsistencies:

- Directory inode numbers pointing to unallocated inodes
- Directory inode numbers greater than the number of inodes in the file system
- Incorrect directory inode numbers for "." and ".."
- Directories that are not attached to the file system

If the inode number in a directory data block references an unallocated inode, `fsck` removes that directory entry.

If a directory entry inode number references outside the inode list, `fsck` removes that directory entry. This condition occurs if bad data is written into a directory data block.

The directory inode number entry for "." must be the first entry in the directory data block. The inode number for "." must reference itself; for example, it must equal the inode number for the directory data block. The directory inode number entry for ".." must be the second entry in the directory data block. Its value must equal the inode number for the parent of the directory entry (or the inode number of the directory data block if the directory is the root directory). If the directory inode numbers are incorrect, `fsck` replaces them with correct values.

Checking file system connectivity

The `fsck` utility checks the general connectivity of the file system. If directories are not linked into the file system, `fsck` links the directory back into the file system by placing it in the lost+found directory.

Running `fsck`

The `fsck` utility is a multipass file system check program, each pass invoking a different phase of `fsck`. After initialization, `fsck` runs successive phases over each file system, checking blocks and sizes, path names, connectivity, reference counts, and cylinder groups. After the final phase, `fsck` performs cleanup operations and exits.

You must be logged in as the superuser to use `fsck`. If you invoke `fsck` without specifying a device name in the command line, it repeats all its phases for all devices. CONVEX recommends that you use `preen` rather than `fsck` in this manner.

The `fsck` utility is an interactive file system check–and–repair program. It uses redundant structural information in the file system to perform consistency checks. For example, `fsck` compares the number of free blocks to the total number of blocks in the file system, minus the number of blocks in use. If the two numbers are not equal, `fsck` displays an error message. Typically, these inconsistencies result from interruption of file system updates, which are performed every time a file is modified.

The `fsck` utility runs in two modes. In the noninteractive mode, it only makes changes to the file system that will always be correct. This mode is used when the operating system is booting. If an inconsistency is found that `fsck` cannot fix, it exits with a nonzero exit status, leaving the system running in single-user mode. In the interactive mode, `fsck` reports the problem and suggests a corrective action; you must decide whether or not to make the corrections.

With the exception of the root (/) file system, unmount a file system to check it. Check the root file system only when the computer is in single-user mode and there is no other activity on the machine.

Note

While in multiuser mode, some file systems will not unmount because daemons are using files in that file system (for example, /tmp).

The format of `fsck` is

`fsck -p` [*file_system*]...

`fsck [-b` *block*`]` [*option* ...] [*file_system*]...

The first form of `fsck`, `fsck -p`, checks a specified file system or a standard set of file systems as specified in the /etc/fstab file. It uses information in this file to inspect groups of disks in parallel, taking advantage of I/O overlap to check file systems as quickly as possible. This form of `fsck` is normally used in the `/etc/rc` script during an automatic reboot. The preen utility can also check disks in this fashion. Refer to the section "The preen utility" in this chapter for more information.

To speed file system checking, `fsck -p` only checks file systems that have the dirty bit set in the superblock. The bit is set when data has been written to a file system since the last *sync to disk*.

The second form uses the following options:

`-b` Use the block specified in *block* as the superblock for the file system. Block 32 is the first alternate superblock for filesystems with block sizes equal to or greater than 16K. Use block 64 for block sizes 32K and greater.

`-c` Inode fields unused prior to ConvexOS V9.0 are zeroed.

`-d` Instructs `fsck` to display diagnostic information while checking file systems.

`-f` Forces `fsck` to look at all file systems, regardless of the setting of the dirty bit. `fsck` uses `-f` as a default unless `-p` is specified.

`-m` The mode given to the lost+found directory if `fsck` needs to create it. The default is 01777.

`-n` Specifies a "no" response to all questions asked by `fsck`. With this option, the file system is not opened for writing.

`-y` Specifies a "yes" response to all questions asked by `fsck`. Use this option with caution— you are giving `fsck` permission to continue even after encountering unlimited difficulties.

You can use `newfs` to determine where alternate super blocks are located. Figure 47 shows a sample `fsck` session.

Figure 47 Sample `fsck` session

```
% fsck /dev/dd0a

/dev/dd0a
Last Mounted on /
Root file system
**Phase 1 - Check Blocks and Sizes
**Phase 2 - Check Pathnames
**Phase 3 - Check Connectivity
**Phase 4 - Check Reference Counts
**Phase 5 - Check Cyl Groups
3163 files, 34968 frags used, 99639 frags free(263 frags, \
12422 blocks) (8k/1k)
%
```

In the example, `fsck` is run without options, and a message is displayed as each phase is completed. If `fsck` encounters an inconsistency, it displays an error message describing the problem.

At the end of the program, a summary message is displayed showing the number of files (inodes), fragments used, and fragments available.

There are seven file system check phases in `fsck` (phase 1B is usually optional), an initialization phase and a clean-up phase. Each phase is summarized below:

- **Phase 1—Check blocks and sizes**—`fsck` checks the inode list. `fsck` may discover error conditions that result from checking inode types, setting up the zero-link-count table, checking inode block numbers for bad or duplicate blocks, checking inode size, and checking inode format.
- **Phase 1B—Rescan for more DUPS**—`fsck` rescans the file system and searches for the inode that previously claimed a particular block. Phase 1B runs only if duplicate blocks (that is, blocks that belong to multiple inodes) are found.
- **Phase 2—Check pathnames**—`fsck` removes directory entries that point to inodes that have noncorrectable error conditions from Phase 1 and Phase 1B. In Phase 2, `fsck` may discover error conditions that result from root inode mode and status, directory inode pointers out of range, and directory entries pointing to bad inodes.

- **Phase 3—Check connectivity**—`fsck` checks directory connectivity seen in Phase 2. In this phase, `fsck` may discover error conditions that result from unreferenced directories and missing or full lost+found directories.
- **Phase 4—Check reference counts**—`fsck` reports messages that result from unreferenced files; a missing or full lost+found directory; incorrect link counts for files, directories, or special files; unreferenced files and directories; bad or duplicate blocks in files and directories; and incorrect total free inode counts.
- **Phase 5—Check cylinder groups**—`fsck` checks the free-block maps. `fsck` looks for errors resulting from allocated blocks in the free-block maps, free blocks missing from free-block maps, and an incorrect total free-block count.
- **Phase 6—Salvage cylinder groups**—`fsck` reconstructs free-block maps. No error messages are produced.

`fsck`

During reboot, running the `fsck` utility on the root file system no longer requires a reboot to recover a corrupt file system. The root file system is automatically remounted and the following message is displayed:

```
/dev/<root>: root filesystem remounted
```

where `<root>` is the name of the root file system device.

In the event that the remount operation fails, `fsck` behaves as it did previously to ConvexOS 11.0 and reboots the system to recover the root file system.

The preen utility

The preen utility runs fsck in a highly optimized manner, checking multiple disk partitions simultaneously. It works similarly to fsck -p, but uses base names of disks to determine which partitions are on the same physical drive and tries to keep all drives busy.

When a system is booted to multiuser mode from the SPU or directly from powerup, preen runs automatically on all file systems before entering multiuser mode. If the machine is booted to single-user mode, you must run preen manually before entering multiuser mode.

One of the greatest advantages of using preen is that system managers no longer need to configure pass numbers in the /etc/fstab file.

The command format for preen is

preen [-L *limit*] [*fsck_options*]

Status returned by preen matches (as closely as possible) that of fsck.

Use the -L option to specify the maximum number (*limit*) of disks for which preen may run concurrent fsck processes. preen with no options starts one fsck process for each disk drive on your system. For systems with large numbers of disks this default condition could lead to memory swapping, cutting down on the efficiency of the preen.

preen passes any fsck options it is given to the fsck processes it starts. Because preen passes these options to all fsck processes, do not give preen the -b option.

Error messages

The `fsck` utility can produce error messages during each phase of execution. The remaining sections in this chapter list these messages.

Error messages are categorized by phase and are presented in the following order:

- Initialization phase
- Phase 1
- Phase 1B
- Phase 2
- Phase 3
- Phase 4
- Phase 5
- Phase 6
- Clean-up phase

A brief description is included with each message. Error conditions that can occur in more than one phase are discussed in the initialization phase.

The following abbreviations are used in `fsck` messages:

BLK	Block number
CG	Cylinder group
DUP	Duplicate block number
DIR	Directory name
MTIME	Time file was last modified
UNREF	Unreferenced

The following single-letter abbreviations are used in `fsck` messages. These abbreviations are replaced by an actual value when the message appears on your screen.

B	Block number
C	Cylinder group
F	File (or directory) name
I	Inode number
M	File mode
N	`fsck` option

O	User ID of a file's owner
S	File size
T	Time file was last modified
X	Link count; number of BAD, DUP, or MISSING blocks; or number of files (depending on context)
Y	Corrected link-count number. Number of blocks in file system (depending on context)
Z	Number of free blocks

Initialization phase

During initialization, tables must be set up and files must be opened. Error conditions in this phase can occur from:

- Command line options
- Memory requests
- Opening files
- The status of files
- File system size checks
- Creation of the scratch file

Note

All initialization errors are fatal when the file system is checked with `fsck -p` or the `preen` utility.

`N option?`

N is not a legal option to `fsck`. Legal options are `-b`, `-c`, `-d`, `-f`, `-m`, `-n`, `-p`, and `-y`. The `fsck` utility is terminated when this error occurs. Refer to the fsck(8) man page for more information.

`cannot alloc NNN bytes for blockmap`

`cannot alloc NNN bytes for freemap`

`cannot alloc NNN bytes for statemap`

`cannot alloc NNN bytes for lncntp`

A `fsck` request for memory for its virtual memory tables failed. The `fsck` utility is terminated when this error occurs.

`Can't open checklist file: F`

The file system checklist file *F* (usually /etc/fstab) cannot be opened for reading. The `fsck` utility is terminated when this error occurs. Check access modes of *F*.

```
Can't stat root
```

`fsck` request for statistics about the root directory (/) failed. The `fsck` utility is terminated when this error occurs.

```
Can't stat F
```

```
Can't make sense out of name F
```

`fsck` request for statistics about the file system *F* failed. When running manually, it ignores this file system and continues checking the next file system given. Check access modes of *F*.

```
Can't open F
```

`fsck` request to open the file system *F* failed. When running manually, it ignores this file system and continues checking the next file system given. Check access modes of *F*.

```
F: (NO WRITE)
```

Either the `-n` flag was specified or an `fsck` attempt to open the file system *F* for writing failed. When running manually, all diagnostics are printed out, but no modifications are made.

```
file is not a block or character device; OK
```

You gave `fsck` a regular file name by mistake. Check the type of file specified.

Possible responses to the `OK` prompt are

YES — Ignore this error condition.

NO — Ignore this file system and continue checking the next file system given.

```
MAGIC NUMBER WRONG
```

```
NCG OUT OF RANGE
```

```
CPG OUT OF RANGE
```

```
NCYL DOES NOT JIVE WITH NCG*CPG
```

```
SIZE PREPOSTEROUSLY LARGE
```

```
TRASHED VALUES IN SUPER BLOCK
```

followed by the message:

```
F: BAD SUPER BLOCK: B
```

```
USE -b OPTION TO FSCK TO SPECIFY LOCATION OF
AN ALTERNATE SUPER-BLOCK TO SUPPLY NEEDED
INFORMATION; SEE fsck(8).
```

The superblock has been corrupted. Select an alternate superblock from among those listed by `newfs` when the file system was created. For file systems with a block size less than 32 kbytes, specifying `-b 32` is a good first choice.

`INTERNAL INCONSISTENCY:` *M*

The `fsck` utility had an internal panic, whose message is specified as *M*.

`CAN NOT SEEK: BLK` *B* `(CONTINUE)`

`fsck` request for moving to specified block number *B* in the file system failed.

Possible responses to the `CONTINUE` prompt are

YES Attempt to continue the file system check. Often, however, the problem will persist. This error condition will not allow a complete check of the file system. Make a second run of `fsck` to recheck this file system. If the block was part of the virtual memory buffer cache, `fsck` terminates with the message "`Fatal I/O error`".

NO Terminate the program.

`CAN NOT READ: BLK` *B* `(CONTINUE)`

`fsck` request for reading specified block number *B* in the file system failed.

Possible responses to the `CONTINUE` prompt are

YES Attempt to continue the file system check. Often, however, the problem will persist. This error condition will not allow a complete check of the file system. Make a second run of `fsck` to recheck this file system. If the block was part of the virtual memory buffer cache, `fsck` terminates with the message "`Fatal I/O error`".

NO Terminate the program.

`CAN NOT WRITE: BLK` *B* `(CONTINUE)`

`fsck` request for writing specified block number *B* in the file system failed. The disk is write-protected.

Possible responses to the `CONTINUE` prompt are

YES — Attempt to continue the file system check. Often, however, the problem will persist. This error condition will not allow a complete check of the file system. Make a second run of `fsck` to recheck this file system. If the block was part of the virtual memory buffer cache, `fsck` terminates with the message "`Fatal I/O error`".

NO — Terminate the program.

Phase 1—Check blocks and sizes

This phase checks the inode list. Error conditions result from

- Checking inode types
- Setting up the zero-link-count table
- Examining inode block numbers for bad or duplicate blocks
- Checking inode size
- Checking inode format

All errors in this phase except `INCORRECT BLOCK COUNT` are fatal if the file system is being checked with the `preen` command.

`CG C: BAD MAGIC NUMBER`

The magic number of cylinder group C is wrong. This usually indicates that the cylinder group maps have been destroyed. When running manually, the cylinder group is marked as needing to be reconstructed.

`UNKNOWN FILE TYPE I=I (CLEAR)`

The mode word of the inode *I* indicates that the inode is not a special block inode, special character inode, socket inode, regular inode, symbolic link, or directory inode.

Possible responses to the `CLEAR` prompt are

YES — Deallocate inode *I* by zeroing its contents. This always invokes the `UNALLOCATED` error condition in Phase 2 for each directory entry pointing to this inode.

NO — Ignore this error condition.

`LINK COUNT TABLE OVERFLOW (CONTINUE)`

An `fsck` internal table containing allocated inodes with a link count of zero has no more room. Recompile `fsck` with a larger value of `MAXLNCNT`.

Possible responses to the `CONTINUE` prompt are

YES — Continue with the program. This error condition will not allow a complete check of the file system. Make a second run of `fsck` to recheck this file system. If another allocated inode with a zero link count is found, this error condition is repeated.

NO — Terminate the program.

B BAD I=*I*

Inode *I* contains block number *B* with a number lower than the number of the first data block in the file system, or greater than the number of the last block in the file system. This error condition may invoke the EXCESSIVE BAD BLKS error in Phase 1 if inode *I* has too many block numbers outside the file system range. This error condition always invokes the BAD/DUP error in Phase 2 and Phase 4.

EXCESSIVE BAD BLKS I=*I* (CONTINUE)

There is more than a tolerable number (usually 10) of blocks with a number lower than the number of the first data block in the file system, or greater than the number of last block in the file system associated with inode *I*.

Possible responses to the CONTINUE prompt are

YES — Ignore the rest of the blocks in this inode and continue checking with the next inode in the file system. This error condition will not allow a complete check of the file system. Make a second run of `fsck` to recheck this file system.

NO — Terminate the program.

HOLD BAD BLOCK?

The system has encountered an inode with a mode field of 017, which is an unknown file type.

YES — Set this inode to indicate a regular file with protection 0600. Set the file size to the size of one fragment.

NO — Leave the inode as is.

B DUP I=*I*

Inode *I* contains block number *B*, and block number *B* is already claimed by another inode. This error condition may invoke the EXCESSIVE DUP BLKS error condition in Phase 1 if inode *I* has too many block numbers claimed by other inodes. This error condition always invokes Phase 1B and the BAD/DUP error in Phase 2 and Phase 4.

EXCESSIVE DUP BLKS I=*I* (CONTINUE)

Other inodes claim more than a tolerable number (usually 10) of blocks claimed by other inodes.

Possible responses to the CONTINUE prompt are

YES Ignore the rest of the blocks in this inode and continue checking with the next inode in the file system. This error condition will not allow a complete check of the file system. Make a second run of `fsck` to recheck this file system.

NO Terminate the program.

```
DUP TABLE OVERFLOW (CONTINUE)
```

An `fsck` internal table containing duplicate block numbers has no more room.

Possible responses to the CONTINUE prompt are

YES Continue with the program. This error condition will not allow a complete check of the file system. Make a second run of `fsck` to recheck this file system. If `fsck` finds another duplicate block, this error condition will repeat.

NO Terminate the program.

```
PARTIALLY ALLOCATED INODE I=I (CLEAR)
```

Inode *I* is neither allocated nor unallocated.

Possible responses to the CLEAR prompt are

YES Deallocate inode *I* by zeroing its contents.

NO Ignore this error condition.

```
INCORRECT BLOCK COUNT I=I (X should be Y)
(CORRECT)
```

The block count for inode *I* is *X* blocks, but should be *Y* blocks. The count is corrected when executing the `preen` utility.

Possible responses to the CORRECT prompt are

YES Replace the block count of inode *I* with *Y*.

NO Ignore this error condition.

Phase 1B—Rescan for more DUPS

When a duplicate block is found in the file system, `fsck` rescans the file system to find the inode that previously claimed that block. The following error condition occurs when the duplicate block is found:

B DUP I=*I*

Inode *I* contains block number *B* that is already claimed by another inode. This error condition always invokes the *BAD/DUP* error condition in Phase 2. You can determine which inodes have overlapping blocks by examining this error condition and the DUP error condition in Phase 1.

Phase 2—Check path names

In this phase, `fsck` removes directory entries pointing to inodes with error conditions in Phase 1 and Phase 1B. Error conditions can occur from

- Root inode mode and status
- Directory inode pointers out of range
- Directory entries pointing to bad inodes

All errors in this phase are fatal if the file system is being checked with the `preen` command.

```
ROOT INODE UNALLOCATED. TERMINATING.
```

The root inode (usually inode number 2) has no allocate mode bits. The program will terminate.

```
NAME TOO LONG F
```

`fsck` found an excessively long path name. This usually indicates loops in the file system name space. This can occur if the superuser made circular links to directories. Remove the circular links.

```
ROOT INODE NOT DIRECTORY (FIX)
```

The root inode (usually inode number 2) is not a directory inode type.

Possible responses to the `FIX` prompt are

YES — Make the root inode type a directory. If the root inode data blocks are not directory blocks, a *very* large number of error conditions will be produced.

NO — Terminate the program.

```
DUPS/BAD IN ROOT INODE (CONTINUE)
```

Phase 1 or Phase 1B found duplicate blocks or bad blocks in the root inode (usually inode number 2) for the file system.

Possible responses to the `CONTINUE` prompt are

YES — Ignore the `DUPS/BAD` error condition in the root inode and attempt to continue the file system check. If the root inode is not correct, this may result in a large number of other error conditions.

NO — Terminate the program.

```
I OUT OF RANGE I=I NAME=F (REMOVE)
```

Inode number *I* of directory *F* is greater than the end of the inode list.

Possible responses to the REMOVE prompt are

YES Remove directory entry *F*.

NO Ignore this error condition.

```
UNALLOCATED I=I OWNER=O MODE=M
SIZE=S MTIME=T DIR F REMOVE [yn]
```

Directory *F* has a directory inode *I* without allocate mode bits. The owner *O*, mode *M*, size *S*, modify time *T*, and directory name *F* are printed.

Possible responses to the REMOVE prompt are

YES Remove directory entry *F*.

NO Ignore this error condition.

```
UNALLOCATED I=I OWNER=O MODE=M
SIZE=S MTIME=T FILE F REMOVE [yn]
```

File *F* has an inode *I* without allocate mode bits. The owner *O*, mode *M*, size *S*, modify time *T*, and file name *F* are printed.

Possible responses to the REMOVE prompt are

YES Remove directory entry *F*.

NO Ignore this error condition.

```
DUP/BAD I=I OWNER=O MODE=M SIZE=S MTIME=T
DIR=F (REMOVE)
```

Phase 1 or Phase 1B found duplicate blocks or bad blocks associated with directory entry *F* and directory inode *I*. The owner *O*, mode *M*, size *S*, modify time *T*, and directory name *F* are printed.

Possible responses to the REMOVE prompt are

YES Remove directory entry *F*.

NO Ignore this error condition.

```
DUP/BAD I=I OWNER=O MODE=M SIZE=S MTIME=T
FILE=F (REMOVE)
```

Phase 1 or Phase 1B found duplicate blocks or bad blocks associated with directory entry *F* and inode *I*. The owner *O*, mode *M*, size *S*, modify time *T*, and file name *F* are printed.

Possible responses to the REMOVE prompt are

YES Remove directory entry *F*.

NO Ignore this error condition.

```
ZERO LENGTH DIRECTORY I=I OWNER=O MODE=M
SIZE=S MTIME=T DIR=F (REMOVE)
```

Size *S* of directory *F* is zero. The owner *O*, mode *M*, size *S*, modify time *T*, and directory name *F* are printed.

Possible responses to the REMOVE prompt are

YES Remove directory entry *F*. This always invokes the BAD/DUP error condition in Phase 4.

NO Ignore this error condition.

```
DIRECTORY TOO SHORT I=I OWNER=O MODE=M
SIZE=S MTIME=T DIR=F (FIX)
```

Size *S* of directory *F* is less than the minimum size directory. The owner *O*, mode *M*, size *S*, modify time *T*, and directory name *F* are printed.

Possible responses to the FIX prompt are

YES Increase directory size to the minimum directory size.

NO Ignore this directory.

```
DIRECTORY CORRUPTED I=I OWNER=O MODE=M
SIZE=S MTIME=T DIR=F (SALVAGE)
```

Directory *F* has an inconsistent internal state.

Possible responses to the SALVAGE prompt are

YES Throw away all entries up to the next 512-byte boundary. This action can throw away up to 42 entries, and should be taken only after other recovery efforts have failed.

NO Skip up to the next 512-byte boundary and resume checking, but do not modify the directory.

```
BAD INODE NUMBER FOR '.' I=I OWNER=O MODE=M
SIZE=S MTIME=T DIR=F (FIX)
```

The inode number for '.' of directory *I* is unallocated.

Possible responses to the FIX prompt are

YES Change the inode number for '.' to be equal to *I*.

NO Leave the inode number for '.' unchanged.

```
MISSING '.' I=I OWNER=O MODE=M SIZE=S
MTIME=T DIR=F (FIX)
```

The first entry of directory *I* is unallocated.

Possible responses to the `FIX` prompt are

YES Make an entry for '.' with inode number equal to *I*.

NO Leave the directory unchanged.

```
MISSING '.' I=I OWNER=O MODE=M SIZE=S
MTIME=T DIR=F
CANNOT FIX, FIRST ENTRY IN DIRECTORY
CONTAINS F
```

The first entry of directory *I* is *F*. The `fsck` utility cannot resolve this problem. Mount the file system and move the entry *F* elsewhere. Then, unmount the file system and run `fsck` again.

```
MISSING '.' I=I OWNER=O MODE=M SIZE=S
MTIME=T DIR=F
CANNOT FIX, INSUFFICIENT SPACE TO ADD '.'
```

The first entry of directory *I* is not '.'.

```
EXTRA '.' ENTRY I=I OWNER=O MODE=M SIZE=S
MTIME=T DIR=F (FIX)
```

Directory *I* has more than one entry for '.'.

Possible responses to the `FIX` prompt are

YES Remove the extra entry for '.'.

NO Leave the directory unchanged.

```
BAD INODE NUMBER FOR '..' I=I OWNER=O MODE=M
SIZE=S MTIME=T DIR=F (FIX)
```

The inode number for '..' of directory *I* does not equal the parent of *I*.

Possible responses to the `FIX` prompt are

YES Change the inode number for '..' to be equal to the parent of *I*.

NO Leave the inode number for '..' unchanged.

```
MISSING '..' I=I OWNER=O MODE=M SIZE=S
MTIME=T DIR=F (FIX)
```

The second entry of directory *I* is unallocated.

Possible responses to the FIX prompt are

YES Make an entry for '..' with inode number equal to the parent of *I*.

NO Leave the directory unchanged.

```
MISSING '..' I=I OWNER=O MODE=M SIZE=S
MTIME=T DIR=F
CANNOT FIX, SECOND ENTRY IN DIRECTORY
CONTAINS F
```

The second entry of directory *I* is *F*. `fsck` cannot resolve this problem. Mount the file system and move entry *F* elsewhere. Then, unmount the file system and run `fsck` again.

```
MISSING '..' I=I OWNER=O MODE=M SIZE=S
MTIME=T DIR=F
CANNOT FIX, INSUFFICIENT SPACE TO ADD '..'
```

The second entry of directory *I* is not '..'.

```
EXTRA '..' ENTRY I=I OWNER=O MODE=M SIZE=S
MTIME=T DIR=F (FIX)
```

Directory *I* has more than one entry for '..'.

Possible responses to the FIX prompt are

YES Remove the extra entry for '..'.

NO Leave the directory unchanged.

Phase 3—Check connectivity

This phase checks the directory connectivity seen in Phase 2. This section lists error conditions resulting from unreferenced directories and missing or full lost+found directories.

UNREF DIR I=*I* OWNER=*O* MODE=*M* SIZE=*S* MTIME=*T* (RECONNECT)

The directory inode *I* was not connected to a directory entry when `fsck` traversed the file system. The owner *O*, mode *M*, size *S*, and modify time *T* of directory inode *I* are printed. When executing the `preen` utility, the directory is reconnected if its size is nonzero; otherwise it is cleared.

Possible responses to the RECONNECT prompt are

YES — Reconnect directory inode I to the file system in the directory for lost files (usually lost+found). This may invoke the lost+found error condition in Phase 3 if there are problems connecting directory inode I to lost+found. This may also invoke the CONNECTED error condition in Phase 3 if the link was successful.

NO — Ignore this error condition. This always invokes the UNREF error condition in Phase 4.

SORRY. NO lost+found DIRECTORY

There is no lost+found directory in the root directory of the file system; `fsck` attempted to create this directory but was unable to do so. This always invokes the UNREF error condition in Phase 4. Check access modes of lost+found. Refer to the fsck(8) man page for more information. This error is fatal if the file system is being checked using the `preen` command.

SORRY. NO SPACE IN lost+found DIRECTORY

There is no space to add another entry to the lost+found directory in the root directory of the file system; `fsck` attempted to expand this directory but was unable to do so. This always invokes the UNREF error condition in Phase 4. Clean out unnecessary entries in lost+found or make lost+found larger. Refer to the fsck(8) man page for more information. This error is fatal if the file system is being checked using the `preen` command.

DIR I=*I1* CONNECTED. PARENT WAS I=*I2*

This is an advisory message: directory inode *I1* was successfully connected to the lost+found directory. The parent inode *I2* of the directory inode *I1* is replaced by the inode number of the lost+found directory.

Phase 4—Check reference counts

This phase checks the link count information seen in Phase 2 and Phase 3. This section lists error conditions resulting from

- Unreferenced files
- Missing or full lost+found directory
- Incorrect link counts for files, directories, symbolic links, or special files
- Unreferenced files, symbolic links, and directories
- Bad and duplicate blocks in files, symbolic links, and directories
- Incorrect total free-inode counts

All errors in this phase can be corrected if the file system is being checked using the `preen` command, except for errors resulting from running out of space in the lost+found directory.

`UNREF FILE I=`*I* `OWNER=`*O* `MODE=`*M* `SIZE=`*S* `MTIME=`*T*
`(RECONNECT)`

Inode *I* was not connected to a directory entry when the file system was traversed. The owner *O*, mode *M*, size *S*, and modify time *T* of inode *I* are printed. When checking the file system using the `preen` command, the file is cleared if either its size or its link count is zero; otherwise it is reconnected.

Possible responses to the `RECONNECT` prompt are

YES — Reconnect inode *I* to the file system in the directory for lost files (usually lost+found). This may invoke the lost+found error condition in Phase 4 if there are problems connecting inode *I* to lost+found.

NO — Ignore this error condition. This always invokes the `CLEAR` error condition in Phase 4.

`(CLEAR)`

The inode mentioned in the previous error condition cannot be reconnected. This cannot occur if the file system is being checked using the `preen` command, because lack of space for reconnecting files is a fatal error.

Possible responses to the `CLEAR` prompt are

YES — Deallocate the inode mentioned in the immediately previous error condition by zeroing its contents.

NO — Ignore this error condition.

```
SORRY. NO lost+found DIRECTORY
```

There is no lost+found directory in the root directory of the file system; `fsck` attempted to create this directory but was unable to do so. This always invokes the CLEAR error condition in Phase 4. Check access modes of lost+found. This error is fatal if the file system is being checked using the `preen` command.

```
SORRY. NO SPACE IN lost+found DIRECTORY
```

There is no space to add another entry to the lost+found directory in the root directory of the file system; `fsck` attempted to expand this directory but was unable to do so. This always invokes the CLEAR error condition in Phase 4. Check size and contents of lost+found. This error is fatal if the file system is being checked using the `preen` command.

```
LINK COUNT FILE I=I OWNER=O MODE=M SIZE=S
MTIME=T COUNT=X SHOULD BE Y (ADJUST)
```

The link count for inode *I*, which is a file, is *X* but should be *Y*. The owner *O*, mode *M*, size *S*, and modify time *T* are printed. When checking the file system using the `preen` command, the link count is adjusted.

Possible responses to the ADJUST prompt are

YES Replace the link count of file inode *I* with *Y*.

NO Ignore this error condition.

```
LINK COUNT DIR I=I OWNER=O MODE=M SIZE=S
MTIME=T COUNT=X SHOULD BE Y (ADJUST)
```

The link count for inode *I*, which is a directory, is *X* but should be *Y*. The owner *O*, mode *M*, size *S*, and modify time *T* of directory inode *I* are printed. When checking the file system using the `preen` command, the link count is adjusted.

Possible responses to the ADJUST prompt are

YES Replace the link count of directory inode *I* with *Y*.

NO Ignore this error condition.

```
LINK COUNT F I=I OWNER=O MODE=M SIZE=S
MTIME=T COUNT=X SHOULD BE Y (ADJUST)
```

The link count for *F* inode *I* is *X* but should be *Y*. The name *F*, owner *O*, mode *M*, size *S*, and modify time *T* are printed. When checking the file system using the `preen` command, the link count is adjusted.

Possible responses to the ADJUST prompt are

YES Replace the link count of inode I with Y.

NO Ignore this error condition.

UNREF FILE I=*I* OWNER=*O* MODE=*M* SIZE=*S* MTIME=*T* (CLEAR)

Inode *I*, a file, was not connected to a directory entry when the file system was traversed. The owner *O*, mode *M*, size *S*, and modify time *T* of inode *I* are printed. When checking the file system using the preen command, this file was not connected because its size or link count was zero; hence it is cleared.

Possible responses to the CLEAR prompt are

YES Deallocate inode *I* by zeroing its contents.

NO Ignore this error condition.

UNREF DIR I=*I* OWNER=*O* MODE=*M* SIZE=*S* MTIME=*T* (CLEAR)

Inode *I*, a directory, was not connected to a directory entry when the file system was traversed. The owner *O*, mode *M*, size *S*, and modify time *T* of inode *I* are printed. When checking the file system using the preen command, this directory was not connected because its size or link count was zero, hence it is cleared.

Possible responses to the CLEAR prompt are

YES Deallocate inode *I* by zeroing its contents.

NO Ignore this error condition.

BAD/DUP FILE I=*I* OWNER=*O* MODE=*M* SIZE=*S* MTIME=*T* (CLEAR)

Phase 1 or Phase 1B found duplicate blocks or bad blocks associated with file inode *I*. The owner *O*, mode *M*, size *S*, and modify time *T* of inode *I* are printed. This error cannot occur when the file system is being checked using the preen command, because it would have caused a fatal error earlier.

Possible responses to the CLEAR prompt are

YES Deallocate inode *I* by zeroing its contents.

NO Ignore this error condition.

```
BAD/DUP DIR I=I OWNER=O MODE=M SIZE=S
MTIME=T (CLEAR)
```

Phase 1 or Phase 1B found duplicate blocks or bad blocks associated with directory inode *I*. The owner *O*, mode *M*, size *S*, and modify time *T* of inode *I* are printed. This error cannot occur when the file system is being checked using the `preen` command, because it would have caused a fatal error earlier.

Possible responses to the `CLEAR` prompt are

YES Deallocate inode *I* by zeroing its contents.

NO Ignore this error condition.

```
FREE INODE COUNT WRONG IN SUPERBLK (FIX)
```

The actual count of free inodes does not match the count in the file system superblock. When checking the file system using the `preen` command, the count is fixed.

Possible responses to the `FIX` prompt are

YES Replace the count in the superblock by the actual count.

NO Ignore this error condition.

Phase 5—Check cylinder groups

This phase checks the free-block maps. This section lists error conditions resulting from allocated blocks in the free-block maps, free blocks missing from free-block maps, and the total free-block count that is incorrect.

```
CG C: BAD MAGIC NUMBER
```

The magic number of cylinder group C is wrong. This usually indicates that the cylinder group maps have been destroyed. When run manually, the cylinder group is marked as needing reconstruction. This error is fatal if the file system is being checked using the `preen` command.

```
EXCESSIVE BAD BLKS IN BIT MAPS (CONTINUE)
```

An inode contains more than a tolerable number (usually 10) of blocks claimed by other inodes or, the number is out of the legal range for the file system. This error is fatal if the file system is being checked using the `preen` command.

Possible responses to the `CONTINUE` prompt are

YES Ignore the rest of the free-block maps and continue execution of `fsck`.

NO Terminate the program.

```
SUMMARY INFORMATION T BAD
```

where *T* can be one or more of the following:

```
(INODE FREE)

(BLOCK OFFSETS)

(FRAG SUMMARIES)

(SUPER BLOCK SUMMARIES)
```

The indicated summary information is incorrect. When checking the file system using the `preen` command, the summary information is recomputed.

```
X BLK(S) MISSING
```

X blocks unused by the file system were not found in the free-block maps. When checking the file system using the `preen` command, block maps are rebuilt.

```
BAD CYLINDER GROUPS (SALVAGE)
```

There are bad blocks in the free-block maps, duplicate blocks in the free-block maps, or blocks missing from the file system. When checking the file system using the `preen` command, the cylinder groups are reconstructed.

Possible responses to the `SALVAGE` prompt are

YES — Replace the actual free-block maps with new free-block maps.

NO — Ignore this error condition.

```
FREE BLK COUNT WRONG IN SUPERBLOCK (FIX)
```

The actual count of free blocks does not match the count in the file system superblock. When checking the file system using the `preen` command, the counts are fixed.

Possible responses to the `FIX` prompt are

YES — Replace the count in the superblock by the actual count.

NO — Ignore this error condition.

Phase 6—Salvage cylinder groups

This phase reconstructs the free-block maps. No error messages are produced.

Clean-up

After a file system has been checked, clean-up procedures are performed. This section lists advisory messages about the file system and modify status of the file system.

```
V files, W used, X free (Y frags, Z blocks)
```

This advisory message indicates that the file system contained *V* files using *W* fragments, leaving *X* fragments free in the file system. Numbers in parentheses breaks the free count down into *Y* free fragments and *Z* free full-sized blocks.

```
***** REBOOT ConvexOS *****
```

This advisory message indicates that the root file system has been modified by `fsck`. If ConvexOS is not rebooted immediately, the work done by `fsck` may be destroyed by the copies of tables ConvexOS keeps. When checking the file system using the `preen` command, `fsck` will exit with a code of 4. The `/etc/rc` script interprets an exit code of 4 by issuing a `reboot` system call.

```
***** FILE SYSTEM WAS MODIFIED *****
```

This advisory message indicates that `fsck` modified the current file system. If this file system is mounted or is the current root file system, halt `fsck` and reboot ConvexOS. If ConvexOS is not rebooted immediately, the work done by `fsck` may be destroyed by the copies of tables ConvexOS keeps.

`fsck`

During reboot, running the `fsck` utility on the root file system no longer requires a reboot to recover a corrupt file system. The root file system is automatically remounted and the following message is displayed:

```
/dev/<root>: root filesystem remounted
```

where `<root>` is the name of the root file system device.

In the event that the remount operation fails, `fsck` behaves as it did previously to ConvexOS 11.0 and reboots the system to recover the root file system.

Performing crash dumps

9

This chapter provides information on using the `crashdump` utility after a system crash. It includes information on

- What a crash dump is
- How to perform a crash dump using
  - The default tape drive
  - A tape drive other than the default
  - The SPU disk
  - The SPU tape drive
- What to do with the crash dump once you have finished making the tape
- How to restart a crash dump if you have tape problems

This chapter does not give details on analyzing crash dumps.

Caution

If you are performing a crash dump, you must do so before rebooting the system after a crash. Rebooting the system initializes memory, destroying the contents of registers that are needed in order to analyze the crash.

Crash dump overview

When a system crashes unexpectedly, you can use its service processor unit (SPU) to create a crash dump. A crash dump is a collection of files that describes the state of a system immediately after a crash.

Systems may crash occasionally due to anomalies that may never reoccur. In this circumstance, a crash dump is not necessary, and rebooting the system is all that is required.

Chronic system problems may exist on rare occasions *that* require further investigation. When a system crashes repeatedly with the message "ConvexOS: FATAL ERROR," or hangs (no panic message, but nothing runs), a crash dump should be recorded to tape. After completing a crash dump under these circumstances, contact the CONVEX Technical Assistance Center (TAC) for a Problem Report (PR) number, which identifies your system problem with the crash dump.

Once a crash dump and its PR number are received by the TAC, specialists can analyze the crash and offer information about how to prevent future occurrences.

Performing crash dumps on a variety of tape drives

This section describes how to prepare for and execute a crash dump. It also describes what to do with the crash dump once it has been recorded onto tape.

There are three ways to record a crash dump. You can use

- A local 9-track tape drive, mt:0, which is the default for `crashdump`
- A local tape drive other than the default
- An mt-format cartridge or DAT SPU tape drive

You can only perform a crash dump on a local tape drive (9-track, rack-mount 3480 cartridge, rack-mount DAT) or a SPU tape drive (mt format or DAT only). Crash dumps to 3480 cartridge or DAT drives that are not rack-mounted, remote tape drives, or to ct-format SPU drives are not supported.

Caution

Do not execute a standalone hardware dump before running `crashdump`. Hardware dumps alter the CPU register state, making it difficult to analyze a potential software problem.

Performing a crash dump using the default, 9-track drive

Step 1 Mount a tape on the mt:0 9-track drive:

- Use a 2400-foot reel tape.
- Place the tape drive online by pressing the **On line** key on the tape drive.
- Make sure the density on the drive is set for 6250.

Step 2 Make sure the keyswitch on the Front Panel is in the LOCAL position.

Step 3 Make sure you have a SPU prompt, which looks like this:

```
(spu)>
```

If you do not have a SPU prompt, press **CTRL-p**.

Step 4 Make sure you are at the /mnt/os directory of the SPU. Enter at the SPU prompt:

```
(spu)> pwd
```

Step 5 If you are not in the /mnt/os directory, change to the /mnt/os directory on the SPU. Enter at the SPU prompt:

```
(spu)> cd /mnt/os
```

Step 6 End execution of all programs other than the standard shell program run by the SPU. Enter at the SPU prompt:

```
(spu)> osclean
```

Step 7 If you are using the default 9-track tape drive, begin the crash dump by entering at the SPU prompt:

```
(spu)> crashdump -H
```

Figure 48 displays the message generated when `crashdump` is started. The -H argument is needed only for hardware crashdumps.

Figure 48 Example `crashdump` output on start

```
(spu)> crashdump -H
Crashdump started Fri Apr 24 13:53:57 CDT 1992
Using hwdump.out
        .
        .
        .
Using /mnt/os/vmunix
```

Step 8 Enter a comment to be included with the crash dump tapes when you are prompted to do so. Provide the following:

- Machine name
- Condition of the machine at crash time
- Problem report number
- Name and number of a person who can be contacted about the crash
- The last time the machine crashed
- Anything else that might be helpful to the TAC.

Figure 49 is an example of a crash dump comment.

Figure 49 Example `crashdump` comment

```
Please enter any comments you have on this crashdump.
Enter ctrl-D on a line by itself to end:
>jupiter (C2400)
>jupiter crashes when the program jinx is running
>PR-1313
>Pat Smith (214) 555-5274
>Last crash was 4/27/92
>^D
The dump will now continue...
```

After `crashdump` begins, status messages are displayed on the SPU console to tell you how the crash dump is progressing.

Step 9 If you are dumping large amounts of memory and need more than one tape, you are prompted when it is necessary to change tapes. Remove the tape that is currently on the drive and replace it with a new tape. Press the **RETURN** key.

Figure 50 shows the prompt to change tapes.

Figure 50 Example mount tape prompt

```
Load next tape (#2 of 2) to continue.
Type <CR> when ready:
```

When changing a tape, ensure that each tape is accurately labeled.

Step 10 When `crashdump` completes writing to tape, it sends a message to the console as in Figure 51.

Figure 51 Example of successful crash dump completion

```
Writing rest of main memory to tape...
mmdump successful
spu>
```

After receiving this message you can remove the tape from the drive. Label the tape with the following information:

- Site name and address
- Problem report number
- Name and number of a person who can be contacted about the crash
- If there is more than one volume, please number the tapes.

Performing a crash dump using a tape drive other than the default

Step 1 Mount a blank tape in the appropriate tape drive (9-track, rack-mount (SCSI) 3480 cartridge, DAT, or SPU). [1]

If you are using a 9-track drive:

- Use a 2400-foot reel tape.
- Place the tape drive online by pressing the **On line** key on the tape drive.
- Make sure the density on the drive is set for 6250.

Step 2 Make sure the keyswitch is in the LOCAL position.

Step 3 Make sure you have a SPU prompt, which looks like this:

```
(spu)>
```

If you do not have a SPU prompt, enter the SPU by pressing **CTRL-p**.

Step 4 Make sure you are at the /mnt/os directory of the SPU. Enter at the SPU prompt:

```
(spu)> pwd
```

Step 5 If you are not in the /mnt/os directory, change to the /mnt/os directory on the SPU. Enter at the SPU prompt:

```
(spu)> cd /mnt/os
```

Step 6 End execution of all programs other than the standard shell program run by the SPU. Enter at the SPU prompt:

```
(spu)> osclean
```

Step 7 Enter at the SPU prompt:

```
(spu)> crashdump -S -H
```

Figure 52 displays the message generated when `crashdump` is started.

[1] To use TLI tape drives, use crashdump's `-t tli` argument.

Figure 52 Example of initial `crashdump` output

```
(spu)> crashdump -S -H
Crashdump started Fri Apr 24 13:53:57 CDT 1992
Using hwdump.out
        .
        .
        .
Using /mnt/os/vmunix
```

Step 8 Enter a comment to be included with the crash dump tapes when you are prompted to do so. Provide the following:

- Machine name
- Condition of the machine at crash time
- Problem report number
- Name and number of a person who can be contacted about the crash
- The last time the machine crashed
- Anything else that might be helpful to the TAC

Figure 53 is an example of a crash dump comment.

Figure 53 Example of a `crashdump` comment

```
Please enter any comments you have on this crashdump.
Enter ctrl-D on a line by itself to end:
>jupiter (C2400)
>jupiter crashes when the program jinx is running
>PR-1313
>Pat Smith (214) 555-5274
>Last crash was 4/27/92
>^D
The dump will now continue...
```

Step 9 Select the alternate drive you are using for the crash dump. Enter an integer that corresponds to one of the drives that is available. In Figure 54 a rack-mounted, 3480 cartridge drive is selected.

Figure 54 Example of selecting an alternate drive for `crashdump`

```
Drive# |  Type    |  Ctlr    | Chassis | CCU | Unit# | Sub-Unit#
=============================================================
     0 | SPUTAPE |    rmt1  |       0 |   0 |     0 |         0
     1 | MTD-001 | MTC-001  |       0 |   0 |     0 |         0
     2 | MTD-207 | MTC-202  |       0 |   0 |     0 |         0
     3 | MTD-207 | MTC-202  |       0 |   0 |     1 |         0

Which tape drive do you wish to use? ('a' to abort):2
Mon Apr 27 13:55:37 CDT 1992
```

Because `crashdump` runs from the SPU, tape drives are listed according to their entries in the SPU's /ioconfig file. Table 3 shows the tape system drive types and their corresponding types in the SPU's /ioconfig file.

Table 3 Tape system and SPU /ioconfig drive types

Tape system drive type	SPU /ioconfig drive type
mt	MTD-001
	MTD-002
	MTD-003
	MTD-004
	MTD-005
	MTD-203
	MTD-204
	MTD-205
ct	MTD-207
dat	MTD-208

After `crashdump` begins, status messages, starting with the date and time, are displayed on the SPU console to tell you how the crash dump is progressing.

Step 10 If you are dumping large amounts of memory and need more than one tape, you are prompted when it is necessary to change tapes. Remove the tape that is currently on the drive and replace it with a new tape. Press the **RETURN** key.

Figure 55 shows the prompt to change tapes.

Figure 55 Example mount tape prompt

```
Load next tape (#2 of 2) to continue.
Type <CR> when ready:
```

When changing a tape, ensure that each tape is accurately labeled.

Step 11 When `crashdump` completes writing to tape, it sends a message to the console as shown in Figure 56.

Figure 56 Example of successful `crashdump` completion

```
Writing rest of main memory to tape...
mmdump successful
spu>
```

After receiving this message you can remove the tape from the drive. Label the tape with the following information:

- Site name and address
- Problem report number
- Name and number of a person who can be contacted about the crash
- If there is more than one volume, please number the tapes.

Performing a crash dump to the SPU disk

Step 1 Make sure the keyswitch is in the LOCAL position.

Step 2 Make sure you have a SPU prompt, which looks like this:

```
(spu)>
```

If you do not have a SPU prompt, enter the SPU by pressing **CTRL-p**.

Step 3 Make sure you are at the /mnt/os directory of the SPU. Enter at the SPU prompt:

```
(spu)> pwd
```

Step 4 If you are not in the /mnt/os directory, change to the /mnt/os directory on the SPU. Enter at the SPU prompt:

```
(spu)> cd /mnt/os
```

Step 5 End execution of all programs other than the standard shell program run by the SPU. Enter at the SPU prompt:

```
(spu)> osclean
```

Step 6 Enter at the SPU prompt:

```
(spu)> crashdump -P <destination directory>
```

Step 7 Enter a comment to be included with the crash dump information when you are prompted to do so. Provide the following:

- Machine name
- Condition of the machine at crash time
- Problem report number
- Name and number of a person who can be contacted about the crash
- The last time the machine crashed
- Anything else that might be helpful to the TAC.

Options for reading these types of crashdumps

After performing a crash dump to the SPU disk, you read its contents using the `crashread` command. Table 4 lists the options to the `crashread` command that are useful in reading crashdumps performed to the SPU disk.

Table 4 Useful `crashread` options

Command	Effect
`crashread -s -f` *<destination_directory>*	Reads crashdump *from* the SPU's directory into which the crashdump was directed *to* the current directory.
`crashread -T [-f` *<tape>*`]`	Use when you tar up the destination directory.
`crashread -D -f` *<directory containing crashdump>*	Useful when you have direct access to the files of a crash dump, for example, when the SPU disk is NFS mounted. This command reassembles the mm.core file.

For more information on using the `crashdump` command, refer to the crashread (8) man page.

Performing a crash dump using the SPU tape drive

Step 1 Mount a blank tape in the SPU tape drive (mt-format or DAT only).

Step 2 Make sure the keyswitch is in the LOCAL position.

Step 3 Make sure you have a SPU prompt, which looks like this:

```
(spu)>
```

If you do not have a SPU prompt, enter the SPU by pressing **CTRL-p**.

Step 4 Make sure you are at the /mnt/os directory of the SPU. Enter at the SPU prompt:

```
(spu)> pwd
```

Step 5 If you are not in the /mnt/os directory, change to the /mnt/os directory on the SPU. Enter at the SPU prompt:

```
(spu)> cd /mnt/os
```

Step 6 Rewind the SPU tape. Enter at the SPU prompt:

```
(spu)> mt rew
```

Step 7 End execution of all programs other than the standard shell program run by the SPU. Enter at the SPU prompt:

```
(spu)> osclean
```

Step 8 Enter at the SPU prompt:

```
(spu)> crashdump -s -H
```

Figure 57 displays the message generated when `crashdump` is started.

Figure 57 Example `crashdump` output on start

```
(spu)> crashdump -s -H
Crashdump started Fri Apr 24 13:53:57 CDT 1992
Using hwdump.out
        .
        .
        .
Using /mnt/os/vmunix
```

Step 9 Enter a comment to be included with the crash dump tapes when you are prompted to do so. Provide the following:

- Machine name

- Condition of the machine at crash time
- Problem report number
- Name and number of a person who can be contacted about the crash
- The last time the machine crashed
- Anything else that might be helpful to the TAC.

Figure 58 is an example of a crash dump comment.

Figure 58 Example `crashdump` comment

```
Please enter any comments you have on this crashdump.
Type CTRL-d on a line by itself to end:
>jupiter (C2400)
>jupiter crashes when the program jinx is running
>PR-1313
>Pat Smith (214) 555-5274
>Last crash was 4/27/92
>^D
The dump will now continue...
```

After `crashdump` begins, status messages are displayed on the SPU console to tell you how the crash dump is progressing.

Step 10 If you are dumping large amounts of memory and need more than one tape, you are prompted when it is necessary to change tapes. Remove the tape that is currently on the drive and replace it with a new tape. Press the **RETURN** key.

Figure 59 on the following page shows the prompt to change tapes.

Figure 59 Example mount tape prompt

```
Load next tape (#2 of 2) to continue.
Type <CR> when ready:
```

When changing a tape, ensure that each tape is accurately labeled.

Step 11 When `crashdump` completes writing to tape, it sends a message to the console as in Figure 60.

Figure 60 Example of successful crashdump completion

```
Writing rest of main memory to tape...
mmdump successful
spu>
```

After receiving this message, you can remove the tape from the drive. Label the tape with the following information:

- Site name and address
- Problem report number
- Name and number of a person who can be contacted about the crash
- If there is more than one volume, please number the tapes.

What to do with crash dump tapes

When the crash dump is complete, label the tape or tapes. Labels on all crash dump tapes should contain the following information:

- Problem Report (PR) number, which is obtained from the TAC
- CPU number of the system that crashed
- Tape volume numbers, if the crash dump is on more than one tape
- Date the crash dump was performed
- Site name
- ConvexOS version number
- Your name and phone number, or the name and phone number of an individual who is familiar with the crash

If a crash dump tape is 9-track, before packaging it to send to the TAC, use a strap, foam block, or similar device to hold the loose end of the tape in place. This reduces the risk of damage during shipment.

After securely packaging the tape or tapes, send them to the following address:

CONVEX Computer Corporation
Technical Assistance Center
P O Box 833851
Richardson TX 75083-3851
USA

Restarting a crash dump

Most crash dump errors occur before `crashdump` begins writing to tape and involve improper use of the tape system. For 9-track drives, for example, if the tape drive is not online, `crashdump` reports the error shown in Figure 61.

Figure 61 Example `crashdump` error

```
Tape not online ... put online and press return
```

Tape system errors that occur before `crashdump` begins writing to tape do not terminate `crashdump`. As the example shows, the `crashdump` utility reports the error and prompts to respond when the error is corrected, and you are ready to continue.

Errors that occur after `crashdump` has started writing to tape cause `crashdump` to terminate. If a crashdump begins writing to a tape and is interrupted and halted by a tape error, do not discard the tape. Put another tape on the machine and restart the crashdump. Send in to the TAC both the first and the second tapes, and make sure the first tape is clearly labeled as such.

Recovering from system crashes

10

Instructions for recovery

After a system crash, it is best to work from the most basic components of the file system outward. Otherwise, corrections made at the higher levels may be undone by programs fixing the lower levels. You must be root to perform these tasks. Follow the order of recovery steps as presented below.

Step 1 Perform a crash dump.
Refer to Chapter 9 for details on how to do this.

Step 2 Perform a `sysreset`.

Step 3 Boot to single-user mode. Enter:

```
(spu)> boot single
```

Step 4 `fsck` the root filesystem before doing any other steps.

Step 5 If a disk failure is not recorded, back-up data for any redundant stripes using that disk will be destroyed by a `preen`. In the unlikely event that a disk drive failed at the same time the machine crashed, perform the following steps. Otherwise, skip to Step 9.

Step 6 In single user mode, enter:

```
% qst
```

Step 7 If the `qst` output reveals that one or more redundant stripes used the failed disk, enter:

```
% getst stxx
```

for each stripe `stxx` that included the failed disk. If the `getst` output shows the disk marked as failed, continue the crash recovery procedure starting at Step 9. Refer to "Using VVM to recover from disk failure" on page 55 for disk failure recovery procedures.

Step 8 If `getst` does not show the disk failure, use `mvst` to move data from the failed disk onto a replacement disk or a hot spare partition. If you replaced the failed disk, enter:

```
% mvst /dev/stxx /dev/dbad
```

where `stxx` is the stripe containing the bad disk and `/dev/dbad` is the device number of the failed disk (and its replacement). To move the data to a hot spare partition, enter

```
% mvst -H /dev/stxx /dev/dbad
```

where `stxx` is the stripe containing the failed disk and `/dev/dbad` is the device number of the failed disk.

`mvst` marks the disk as failed, so once you initiate the `mvst`, it is safe to run `preen` on your file systems.

Step 9 Run `preen` to check the file systems. There may be user, system, or audit files in the process of being built when the system crashes. Although that data may be lost, `preen` can recover some of those files in the lost+found directory of the file system and can fix basic file system problems. Enter:

```
% preen
```

Step 10 Boot the system to multiuser mode. Press **CTRL-D**.

Users can now start using the system.

Line printer system error messages

A

The ConvexOS line printer system may produce any of the following error messages or warnings. These messages are categorized according to the printer utility that produces them, as listed below:

- `lpc`
- `lpd`
- `lpmv`
- `lpq`
- `lpr`
- `lprm`

If you encounter error messages that are not described, this may indicate a serious internal error. In that case, please contact the CONVEX Technical Assistance Center (TAC).

Throughout this appendix, the term *printer* is the name of a printer from the /etc/printcap file.

lpc error messages

The following messages are produced by the lpc utility:

```
cannot examine spool directory
```

Error messages that begin with "cannot" are usually caused by incorrect ownership or protection mode of the lock file, spooling directory, or the lpc utility. Run the /etc/verify command to determine if any files or directories do not have the proper characteristics.

```
couldn't start printer
```

The connection to the line printer daemon, lpd, on the local machine failed. This usually means that the master printer daemon started during the boot process has died or is hung. To determine whether the daemon is running, enter:

```
% /etc/lpc status printer
```

You can also determine whether the daemon is running with the following command:

```
% ps ax | grep lpd | grep -v grep
```

If no /usr/lib/lpd process is displayed, restart the daemon using the lpc restart command

```
% /etc/lpc restart printer
```

or enter

```
% /usr/lib/lpd
```

Another possibility is that the lpc program does not have its owner and group set to daemon, or the setuid and setgid are not set. This can be checked with the command:

```
% ls -lg /usr/etc/lpc
```

The following message will be displayed if everything is set correctly:

```
-r-sr-sr-x 1 lpr lpr 143807 Jun 29 16:57
/usr/etc/lpc
```

lpd error messages

The lpd process can log many different messages using syslogd. Most of these messages are about files that cannot be opened and usually indicate that the /etc/printcap file or the protection modes of the files are incorrect. Files may also be inaccessible if users manually manipulate the line printer system by bypassing the lpr program.

In addition to messages generated by lpd, any one of the filters that lpd creates may log messages using syslogd to the error file specified in the lf field of the /etc/printcap file.

lpmv error messages

The following message is produced by the lpmv utility:

```
permission denied
```

A user attempted to move file(s) that are not owned by the user.

lpq error messages

The following messages are produced by the lpq utility:

```
no space on remote; waiting for queue to
drain
```

This message means that there is insufficient disk space on the remote machine. If the file is large enough, there may never be enough space on the remote machine (even when the queue is empty). To correct this problem, move the spooling queue or create more disk space on the remote machine.

printer `is ready and printing`

The lpq utility checks to see if a daemon exists for *printer* and prints the status file located in the spooling directory. If the daemon is hung, use the lpc utility to abort the current daemon and start a new one.

`sending to` *host*

Files are being transferred to the remote machine *host*.

`waiting for` *host* `to come up`

This message means that a daemon is trying to connect to a remote machine named *host* to send files in the local queue. Use the /usr/etc/ping command to see if the remote machine is up. If the remote machine is up, the daemon on the remote machine has probably died or is hung and must be restarted.

```
waiting for printer to become ready (offline?)
```

The daemon could not open the printer device. This happens for several reasons; the most common are that the printer is offline, powered off, or physically disconnected. Other possibilities are that the printer is out of paper or the paper is jammed. Not all printers supply enough information to indicate whether the printer is having trouble or is merely offline. The only way to tell what is actually wrong (without physically going to the printer) is through the error code returned by the device driver.

Another possible cause of this message is that a process, such as an output filter, has already opened the device and has died or is hung. If this is the case, you must kill the offending process and restart the printer with the `lpc` utility.

```
warning: no daemon present
```

The line printer daemon that controls the queue does not exist. This message occurs when the daemon has unexpectedly died. Check the error log file and the `syslogd` files for diagnostics from the daemon. To restart the daemon, enter:

```
% /etc/lpc restart printer
warning: printer is down
```

The printer has been made unavailable by the `lpc` utility.

lpr error messages

The following messages are produced by the `lpr` utility:

`lpr: printer: jobs queued, but cannot start daemon`

The connection to the line printer daemon, `lpd`, on the local machine failed. This usually means that the master printer daemon started during the boot process has died or is hung. To determine whether the daemon is running, enter:

```
% /etc/lpc status printer
```

You can also determine whether the daemon is running with the following command:

```
% ps -ax | grep lpd | grep -v grep
```

If no `/usr/lib/lpd` process is displayed, restart the daemon using the `lpc restart` command:

```
% /etc/lpc restart printer
```

or enter

```
% /usr/lib/lpd
```

Another possibility is that the `lpr` program does not have its owner and group set to daemon, or the setuid and setgid are not set. This can be checked with the command:

```
% ls -lg /usr/ucb/lpr
```

The following should be displayed if everything is set correctly:

```
-r-sr-sr-x 1 lpr lpr 124725 Jun 29 16:57
/usr/ucb/lpr
```

`lpr: printer: printer queue is disabled`

The printer queue is disabled. Jobs can be sent to the queue, but nothing will be printed until the queue is enabled with the command:

```
% /etc/lpc enable printer
```

`lpr: printer: unknown printer`

The *printer* was not found in the printcap file. Usually this is a typing mistake; however, it may indicate a missing or incorrect entry in the /etc/printcap file.

`lpr: cannot access filename`

`lpr` cannot locate a file with the provided filename.

lprm error messages

The following messages are produced by the `lprm` utility:

`lprm: printer: cannot restart printer daemon`

The connection to the line printer daemon, `lpd`, on the local machine failed. This usually means that the master printer daemon started during the boot process has died or is hung. To determine whether the daemon is running, enter:

```
% /etc/lpc status printer
```

You can also determine whether the daemon is running with the following command:

```
% ps ax | grep lpd | grep -v grep
```

If no `/usr/lib/lpd` process is displayed, restart the daemon using the `lpc restart` command

```
% /etc/lpc restart printer
```

or enter

```
% /usr/lib/lpd
```

Another possibility is that the `lprm` program does not have its owner and group set to daemon, or the setuid and setgid are not set. This can be checked with the command:

```
% ls -lg /usr/ucb/lprm
```

The following message will be displayed if everything is set correctly:

```
-r-sr-sr-x 1 lpr lpr 147982 Jun 29 16:57
/usr/ucb/lprm
```

```
hostname: cfA792sourcehost: Permission denied
hostname: cfA792sourcehost: Permission denied
```

You do not have permission to dequeue the file or user you specified.

Reporting problems

B

The CONVEX Technical Assistance Center (TAC) is staffed by technical specialists who can address diverse questions and problems arising in a supercomputing environment. The TAC recommends using the `contact` utility to inquire about hardware, software, or documentation. `contact` is an interactive program that helps the TAC track reports and route them to the CONVEX personnel most qualified to handle them.

This appendix describes

- Prerequisites for using `contact`
- Tips for using `contact`
- Step-by-step procedure for using `contact`

Prerequisites for using `contact`

The `contact` utility requires:

- UNIX-to-UNIX Communication Protocol (UUCP) connection to the TAC
- Full path name of the program or utility about which you have an inquiry
- Version number of the program or utility about which you have an inquiry

UUCP connection

Before using `contact`, ask your system manager if your site has a UUCP connection to the TAC. A UUCP connection allows files to be copied from one UNIX-based system to another. The `uucp` (UNIX-to-UNIX copy) command relies on either a dial-up or hard-wired UUCP communication line.

Using `which` to find a program's path name

To determine the full path name of a program or utility, use the `which` command, which has the following syntax:

`which` *program*

program is the name of the program whose path you need. Figure 62 illustrates the use of `which` to find the full path name of a fictitious program called `filefix`.

Figure 62 Using the `which` command

```
% which filefix
/bin/filefix
%
```

In this example, the full path name of `filefix` is /bin/filefix.

If you use `csh` or `ksh`, you can use the `whence` command to find a program path name instead. `whence` works the same as `which`, but faster.

For more information on the `which` or `whence` command, refer to the which(1) or whence(1) man page, respectively.

Using `vers` to find a program's version number

To determine the version number of a program or utility, use the `vers` command, which has the following syntax:

`vers` *path*

path is the full path name of the program or utility whose version number you need. Figure 63 illustrates the use of `vers` to find the version number of the program `filefix`.

Figure 63 Using the `vers` command

```
% vers /bin/filefix
/bin/filefix: 10.0
%
```

In this example, the version number of `filefix` is shown to be 10.0.

For more information on the `vers` command, refer to the vers(1) man page.

Tips for using `contact`

This section lists tips to help you use `contact` efficiently. In particular, this explains

- Creating a `.contact` file
- Suspending a `contact` session
- Moving within `contact` from one prompt to another
- Using tilde-escape sequences within `contact`
- Aborting your `contact` report
- Submitting your aborted report

Creating a .contact file

When you invoke `contact`, it first prompts for your name, title, phone number, and company name. You can, however, create a `.contact` file to skip this first prompt. `contact` will look for a `.contact` file and use the information in that file automatically.

Follow these steps to create a `.contact` file:

1. Create a `.contact` file in your home directory.
2. Enter your name, job title, phone number, and company name, each on a new line.

In Figure 64 is an example of a `.contact` file viewed using the `cat` command

Figure 64 Example of a `.contact` file (viewed with the `cat` command)

```
% cat .contact
Chris Smith
Programmer
(214) 900-2000
Jupiter Corporation
%
```

Suspending your `contact` session

Sometimes it is necessary to suspend your `contact` session and return to your shell (for instance, to find your program path

name or version number). To suspend your `contact` session, press **CTRL-z**.

To return to the `contact` session, type `fg`. Using **CTRL-z** and the `fg` (foreground) command, you can switch between `contact` and your shell. You cannot, however, use **CTRL-z** and `fg` to switch back and forth if you use Bourne shell (`sh`).

Moving to another prompt

The `contact` utility prompts for information pertinent to your hardware, software, or documentation question. Some prompts require one-line responses; to move to the next prompt, press **RETURN**. Other prompts require more than a one-line response; to move to the next prompt, press **CTRL-d**.

Tilde-escape sequences

The `contact` utility treats input beginning with a tilde (~) as a special sequence. The character following the tilde is considered a request for a special function.

You cannot use tilde-escape sequences when listing file names to include in your report, as described in Step 13 on page 165.

Any other time you can use the following tilde sequences within `contact`:

~e	Edit your report using your default editor (defined in your EDITOR environment variable)
~h	Help by displaying a list of available tilde-escape sequences
~p	Print your `contact` report to the terminal screen
~r *filename*	Read the contents of a specified file into a response to the current prompt, where *filename* is the name of the file you wish to specify. Some prompts require only a one-line response. This tilde-escape sequence works only for prompts that allow more than a one-line response.
~~	Insert a single tilde as the first character in the line. This is in case you need to use a tilde as part of your response and not as an escape sequence.

Aborting your report

To abort a `contact` report, either press the interrupt key (usually **CTRL-c**) or choose the `abort` option when prompted by the `contact` utility as described in Step 14 on page 166. Using **CTRL-c** to abort does not save the contents of the report. Using the `abort` option saves the contents of the report in a file named ~/dead.report.

Submitting your dead.report file

After you abort a `contact` report (using option 4, as shown in Step 14 on page 166), `contact` saves the report in a file named ~/dead.report. If you specify the `-r` option when you next invoke `contact`, it automatically merges the contents of the ~/dead.report file from your previous report into your current `contact` report. For more information on how to use the `-r` option, refer to Step 3 on page 159.

Using `contact`

Before using `contact`, refer to the previous section, "Prerequisites for using `contact`," which gives you important information you must know in order to use `contact` effectively.

Step 1 Before starting the `contact` utility, you need to know the full path name of the program on which you are reporting. If you do not know the full path name, determine it using the `which` command. See Figure 62 on page 154 for an example of the which command.

Step 2 You also need the version number of the product for which you are submitting the report. You can determine the version number using the `vers` command. You must supply the full path name of the program in question. See page 155 for an example of the `vers` command.

Step 3 Invoke `contact`.

The `contact` utility has the following syntax:

```
contact option
```

where *option* can only be

```
-r
```

which includes the ~/contact.dead file of your previous report with your current `contact` report. For more information on this option, refer to the section "Creating a .contact file" on page 156.

`contact` can be invoked as shown in Figure 65. After invoking `contact`, the system responds with a welcome message, reports

information on the system you are logged in from and asks if you wish to report a problem for a different machine.

Figure 65 Beginning a contact report

```
% contact
Welcome to contact version 0.24 (93/07/19)

hostname (serial number 99) has been identified as an
internal Convex machine, or this contact report is being
submitted for another CPU.

Would you like to specify a different machine (yes | no)?
>
```

If you do not wish to specify another machine answer "no" at the prompt and go to Step 4.

If you do wish to specify another machine answer "yes" at the prompt. contact then asks for the new CPU serial number and the new hostname, as shown in Figure 66.

Figure 66 Sample contact report for another machine

```
% contact
Welcome to contact version 0.24 (93/07/19)

hostname (serial number 99) has been identified as an
internal Convex machine, or this contact report is being
submitted for another CPU.

Would you like to specify a different machine (yes | no)?
> yes
What is the new CPU serial number?
> 999999
What is the hostname for CPU #999999?
> new_hostname
Machine ID set to new_hostname, CPU 999999
```

Step 4 contact then asks a series of questions about you and your hardware, software, or documentation question, as shown in Figure 67, unless you have a .contact file in your home directory. If you have a .contact file in your home directory, contact skips this inquiry. (Refer to the section "Creating a .contact file" on page 156.)

Figure 67 Beginning a `contact` report without a .contact file

```
% contact
Welcome to contact version 0.21 (92/02/10)

hostname (serial number 99) is an internal Convex machine.
Would you like to specify a different machine (yes | no)?
> no

Enter your name, title, phone number, and corporate name (^D when finished)
> Chris Smith
> Programmer
> (214) 900-2000
> Jupiter Corporation
> ^D
```


Figure 68 illustrates how the system responds if you have a .contact file in your home directory. If you have a .contact file, go to Step 5.

If you do not have a .contact file—to use `contact` effectively—you must then provide the following information:

1. Your name, title, phone number, and corporate name.
2. Name of the product with which you are having a problem.
3. Version number of the product with which you are having a problem.
4. One-line summary of the problem.
5. Detailed description of the problem.
6. Priority of the problem.
7. Instructions on how to reproduce the problem.
8. Comments about the problem.
9. Comments about the documentation relating to the problem.
10. Whether files are included in the `contact` report and, if so, the full path names of these files.
11. How you would like to complete your `contact` session.

Step 5 Enter the product name.

The `contact` utility next prompts for the name of the product with which you are experiencing a problem, as shown in Figure 68. Enter the name of the product.

Figure 68 Beginning a `contact` report with a `.contact` file

```
Enter the name of the product involved
> filefix
```

Step 6 Specify a program version number.

The `contact` utility prompts for the version number of the product with which you are experiencing a problem, as shown in Figure 69.

Figure 69 Prompt for product version number

```
Enter the version number (in the form X.X or X.X.X.X) of the product
> 9.0
```

If you do not know the version number, press **CTRL-Z** to suspend the session and refer to the section "Using vers to find a program's version number" on page 155. After you have determined the proper version number, use the `fg` command to return to the `contact` session and enter the version number in the form *X.X* or *X.X.X.X*, such as

```
9.0
```

or

```
9.0.0.1
```

Step 7 Enter a one-line problem summary.

The `contact` utility prompts for a one-line summary of the problem, as shown in Figure 70.This summary is the subject

header in any further correspondence regarding your problem, make it as descriptive as possible in one line.

Figure 70 Prompt for a short problem summary

```
Enter a short (1 line) summary of the problem
> Trouble suspending filefix
```

Step 8 Enter a detailed problem description.

The `contact` utility prompts for a detailed description of the problem, as shown in Figure 71.

Figure 71 Prompt for a detailed description of the problem

```
Enter a detailed description of the problem (^D to terminate)
> After entering filefix, I have trouble suspending the session if I want
to do other tasks.
> ^D
```

When writing your problem description, please make it as complete as possible. Include source code and a stack backtrace when possible. (Refer to the adb(1) or csd(1) man pages for information on obtaining a stack backtrace.) The more information you provide, the quicker the TAC can isolate and solve your problem.

When you have completed your description, press **CTRL-d**.

Step 9 Enter a problem priority.

`contact` prompts for the priority of your problem. The priority of a problem indicates the impact your problem has on your work. Figure 72 shows how `contact` prompts for priority

levels. Select your problem priority by entering the number associated with the priority.

You must enter the number associated with your problem priority.

Figure 72 Prompt for problem priority

```
Enter a problem priority, based on the following:
1) Critical    - work cannot proceed until the problem is resolved.
2) Serious     - work can proceed around the problem, with difficulty.
3) Necessary   - problem has to be fixed.
4) Annoying    - problem is bothersome.
5) Enhancement - requested enhancement.
6) Informative - for informational purposes only.

Priorities 1-3 are reserved for functionality problems. For errors,
omissions, and typos in man pages or documentation, please use priority 4.
> 4
```

Step 10 Give instructions how to reproduce the problem.

`contact` prompts for instructions on how to reproduce the problem, as shown in Figure 73.

Figure 73 Prompt for instructions on how to reproduce the problem

```
Enter instructions by which the problem may be reproduced (^D to terminate)
> 1. Enter filefix myfile.c
> 2. Suspend by entering CTRL-s
> ^D
```

Please include the command syntax and options you used and anything else you did to make the program run.

Step 11 Give applicable comments.

The `contact` utility prompts for any other pertinent comments, as shown in Figure 74. Please include all relevant information.

Figure 74 Prompt for additional, applicable comments

```
Enter any comments that are applicable(^D to terminate)
> Perhaps I am using the wrong command?
> ^D
```

Step 12 Offer suggestions for documentation and support.

The contact utility prompts for suggestions regarding documentation supporting the product, as shown in Figure 75.

Figure 75 Prompt for suggestions or comments

```
(Optional) Do you have any suggestions or comments on the documentation
that you referenced when you were trying to resolve your problem (for
example, additions corrections, organization, accessibility)? (^D to
terminate)
> A command summary or quick-reference for filefix would be helpful.
> Documentation could be revised for this.
> ^D
```

Please indicate whether the documentation could be revised to address the problem.

Step 13 Indicate additional files.

The contact utility prompts for names of files necessary to reproduce the problem.

If you have files that can be included with your report, enter "yes" and enter the related file names, as shown in Figure 76.

List file names one per line. After entering your last file name, press **CTRL-d**. Tilde-escape sequences are not recognized in your file listing. A tilde (~) in this section indicates your home directory.

Figure 76 Including additional files in your contact report

```
Are there any files that should be included in this report (yes | no)?
> yes
Please enter the names of the files, one to a line (^D to terminate)
> myfile.c
> ^D
```

If you do not have any files to include with your report, enter "no" and you will be prompted for the next response.

If files specified are small text files, they are automatically included in the contact report. If the files are too large to be included in this report, contact gives further instructions on how to submit these files.

To specify a directory, combine directory files into a single file using the tar command (refer to the tar(1) man page for further

information) or enter each file name in the directory on a single line in the `contact` report.

Step 14 To finish your `contact` report, you are given the choice to review, edit, submit, or abort the report, as shown in Figure 77.

You must enter the number associated with your selection.

Figure 77 Prompt to review, edit, submit, or abort your `contact` report

```
Please select one of the following options:
1) Review the problem report.
2) Edit the problem report.
3) Submit the problem report.
4) Abort the problem report.
> 3
```

The options listed in Figure 77 indicate the following:

Review	Review the text of the `contact` report. You are then prompted again to select an option.
Edit	Edit the text of the `contact` report. If you choose to edit the report, `contact` opens your default text editor.
Submit	Send the report to the CONVEX TAC. The TAC notifies you within 48 hours that your report has been received. Choosing this option exits the `contact` utility and returns you to your shell.
Abort	Save the text of the report in a file named ~/dead.report (in your home directory). Choosing this option exits `contact` and returns you to your shell.

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

ORDER NUMBER
DSW-031

DOCUMENT NUMBER
710-011830-005