Managing ConvexOS Configuration Guide

Third Edition

CONVEX COMPUTER CORPORATION

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

Managing ConvexOS: Configuration Guide

Order No. DSW-030

Third Edition
July 1992

CONVEX Press
Richardson, Texas
United States of America

Managing ConvexOS: Configuration Guide

Order No. DSW-030

Printed in the United States of America

Revision information for

Managing ConvexOS: Configuration Guide

Edition	Document No.	Description
Third	710-001430-211	Released with ConvexOS V10.1, July 1992.
Second	710-001430-210	Released with ConvexOS V10.0, December 1991.
First	710-001430-209	Initially released with ConvexOS V9.0, October 1991.

Contents

Figures

Using this guide

1 Security considerations

2 Adding devices

3 Setting up the disk system

4 Scheduling file system backups

5 Setting up the line printer system

6 Setting up a UUCP connection

7 Setting up user accounts

8 Setting up the accounting system

9 Setting quotas on disk space use

10 Setting up `sendmail`

11 Setting up the notesfile system

12 Setting up log files

13 Setting up online man pages

14 Granting operator-class privileges

15 Customizing kernel boot-time parameters

16 Generating system images

17 Configuring the `contact` utility

Tables

Using this guide

Managing ConvexOS: Configuration Guide describes the configuration tasks the system manager must perform to configure system resources, such as customizing boot-time parameters, setting up the disk, tape, and line printer systems, and adding new users.

Organization

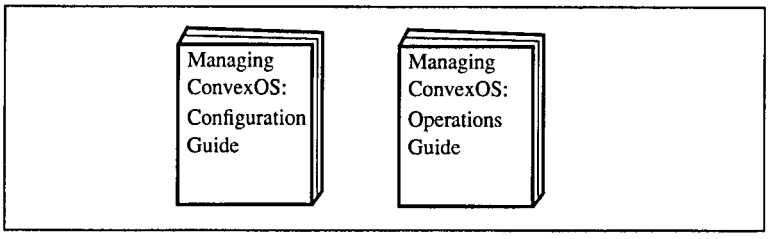
Managing ConvexOS is a multilayered task. The information required to manage ConvexOS can be divided into two categories:

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

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

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

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



This volume, 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

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.

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* contains 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 index in this guide is a master index that covers the following books:

- *ConvexOS Tape System Manager's Guide*
- *ConvexOS Tape System Operator's Guide*
- *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/convex
- /usr/adm
- /usr/ucb
- /ucb/bin

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 functions similarly to `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 the 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 Man Pages for Users* (DSW-331), *ConvexOS Man Pages for Programmers* (DSW-332), *ConvexOS Man Pages for System Managers* (DSW-333). This set of three books is the standard reference for the ConvexOS operating system.
- *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 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 the CONVEX C3800 Series machines
- *Managing ConvexOS: Operations Guide* (DSW-031), a guide for maintaining ConvexOS

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.

Security considerations 1

Many of the configuration tasks described in this manual require you to make and implement decisions on security issues. This chapter introduces security as a topic and briefly summarizes some of the methods used to implement security decisions. However, the actual details for implementation are included in the appropriate chapters throughout this manual.

This chapter describes:

- How to protect access to the system
- How to protect system and user files
- How to detect when someone unsuccessfully attempts to access the system or files in the system

For more information on security issues, see "The transitive property of insecurity" in the *ConvexOS Tutorial Papers*.

Protecting access to the system

Protecting access to the system involves controlling physical access, login access, and remote access.

Protecting physical access

Part of protecting your system is inhibiting physical access to the system by unauthorized users. One way to do this is by restricting entry into the building or room containing the system and back-up media.

If your users use a C shell, you can protect against unattended terminals by using the `autologout` option. This option automatically logs out users whose shells have been idle for a specified time. It is highly recommended as an addition to root's .cshrc file. To enable this option, include the following line in /etc/login or the .cshrc file of each user:

```
set autologout = min
```

min is the number of minutes the shell can be idle before automatic logout. The default autologout values are 60 minutes for user shells and 15 minutes for root shells. See Chapter 7, "Setting up user accounts," on page 161, for details on changing the .login and .cshrc files.

Protecting UUCP or dial-in access

The uucp program copies files from system to system, often across phone lines. Because it is designed to provide access to your system by remote users, it is a potential security risk unless carefully administered.

The easiest way to avoid unauthorized use through phone lines is to eliminate the phone lines. However, that restricts users to working on-site and eliminates dial-in access to the worldwide USENET. USENET is a subset of the UUCP network that connects thousands of systems over dial-in phone lines that exchange news items worldwide.

The best approach to UUCP security requires assigning UUCP sites their own UUCP logins and passwords. The passwords also establish an audit trail to be used in case the system is compromised.

When carefully configured, the UUCP system is difficult to compromise. Use the following requirements to protect it:

- Require remote sites, in the same way as local users, to sign on with a login name and password.
- Require remote sites to provide the name of the system they are calling from. The system checks this name against a list of authorized remote sites before allowing the user to log in.
- Require a callback protocol for remote sites by specifying the callback option in /usr/lib/uucp/USERFILE. When a call is received, the local system terminates the connection and immediately calls back the remote host.
- Permit remote sites access to only a few programs after logging in.
- Make a UUCP administrator owner of files located in the /usr/lib/uucp directory and secure them from access by unauthorized users. When your system software is initially loaded, these files have the proper permissions. Do not alter these permissions when you make changes to files in this directory.
- In addition to the uucp program, you can set up a dial-in password for your system. CONVEX recommends changing the dial-in password monthly. Refer to Chapter 2, "Adding devices," on page 21, for implementation details.

Refer to Chapter 6, "Setting up a UUCP connection," on page 143, for details on implementing these requirements.

Protecting login access

If an unauthorized user gains access to a terminal line connected to the system, only the lack of a valid login name and password prevents entry into the system. Unfortunately, login names are easily deduced because they are almost always the user's first name, last name, or initials. Consequently, ConvexOS requires a password before it grants a user access to the system.

The primary characteristic of a secure password (that is, one that resists detection) is that it is not obvious or easily derived. Instruct your users to follow these minimum guidelines when selecting passwords:

- Do not use passwords based on family names, maiden names, initials, phone numbers, or social security numbers.
- Never use a word from any dictionary, foreign language or any other, unless it is altered by a slight misspelling or by mixing one or more punctuation marks with the characters.
- Change passwords frequently, especially for root.
- Encourage users to change their passwords on a regular basis.

As system administrator you can invoke two types of password restrictions to help secure against unauthorized access: typing restrictions and password aging. Password restrictions increase security at your site and ensure that users participate in password security by selecting secure passwords and by changing those passwords regularly. You can set either or both of these restrictions for each individual user.

If you specify typing restrictions for a user, any password selected must:

- Contain at least six characters
- Contain at least two alphabetic characters and one numeric or special character
- Not be the user's login name or any rotated permutation of it
- Differ from the previous password by a minimum of three characters

If you specify password aging restrictions for a user, you can enforce the following rules:

- A password must remain unchanged for a minimum number of weeks. This means users cannot change back to their original password immediately after being forced to select a new one.

- A password remains valid for a maximum number of weeks. When the password is no longer valid, the user is prompted to set a new password, and must set a new password in order to login.
- Temporary passwords, which are valid for one login, and other special passwords are possible using special age codes.

Passwords are recorded in the /etc/passwd file in an irreversibly encrypted form. Refer to Chapter 7, "Setting up user accounts," on page 161, for details on implementing password restrictions.

Protecting access to files

Each file and directory in the ConvexOS system has attributes that specify who can access it and the degree of access allowed.

File access can be:

Read	Allows users to read a file. This includes copying the file.
Write	Allows users to modify the contents of a file. This includes truncating and appending the file.
Execute	Allows users to execute a program or script file.

ConvexOS provides the following types of directory access:

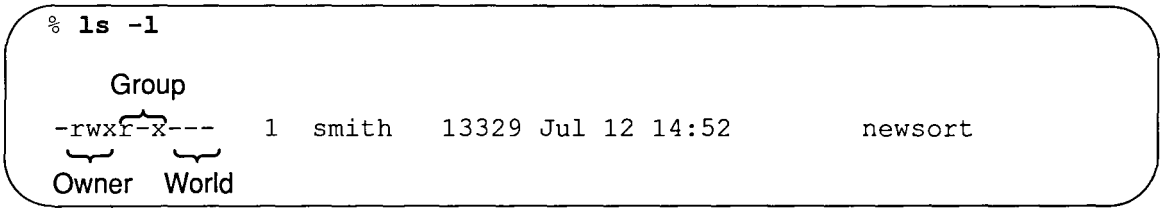
Read	Allows users to list a directory's contents.
Write	Allows users to alter the contents of a directory, such as create files, delete files, or rename files, regardless of the type of access assigned to the file.
Execute	Allows users to use the directory. This includes searching the directory and its subdirectories. Without execute permission, you are not able to use the files in the directory, even if you have read and write permissions on the directory.

Permission to read, write, or execute a file or directory can be extended to one or more of the following classes of users:

User	Owner of the file.
Group	Members of the group to whom the file belongs. Typically this is a group to which the owner of the file belongs. By default, group is set to the group of the directory in which the file appears.
Others	All other users of the system. That is, not the owner, and not the users who belong to the group.

You can view the file-access permissions extended to user, group, and others for a file using the `ls` command with the `-l` argument. See Figure 2 for sample `ls -l` output.

Figure 2 Sample `ls -l` output



Access permissions are expressed symbolically in this output. The symbolic assignments for file access levels are:

`r` = read

`w` = write

`x` = execute

The second, third, and fourth characters express the access permission set for owner; the fifth, sixth, and seventh characters express the access permission set for group; and the eighth, ninth, and tenth characters express the access permission set for other.

The first position in a permission set indicates whether or not the owner class has read access, the second position indicates write access, and the third position indicates execute access. If a dash (-) appears in any position, the access is not granted. For example, if a dash appears in the first position of a set, read access is not granted.

In the sample output, the owner has read, write, and execute access, group has read and execute access, and everyone else has no access.

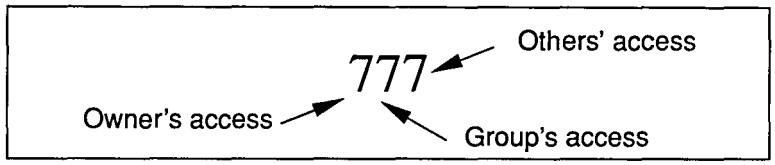
If the user is the owner, the system checks only the owner permission set. If the user belongs to the file's group, the system checks only the group permission set. If the user is anyone else, the system checks the other permission set.

For example, assume that permission is set for others to allow read access but permission is set for group to deny read access. If you belong to the group, you are not permitted to read even though everyone else can.

Default file access

File access permissions are set for each file and directory when it is created, but are expressed in a set of three octal digits, for example, 777. The first digit describes the owner's access, the second digit describes the group's access, and the third digit describes all others' access, as shown in Figure 3.

Figure 3 File access permission fields



Each access level (that is read, write, and execute) is assigned an octal number. The octal assignments are:

4 = read access

2 = write access

1 = execute access

If a file's access permissions are 421, the owner has read access, the group has write access, and others have execute access.

To grant more than one type of access to a user class, for example read and write access, you must add the octal digits assigned to each access type. For example, to grant read and write access, set the access permission to 6:

```
   read access =  4
+  write access = 2
   sum =          6
```

When a user creates a new file or directory, the file or directory is granted access permissions based on the specified umask value. The umask is a set of values that take away access permissions. The umask value is expressed in three octal digits that correspond to the file access settings.

For example, if you set the umask value to 222, you would be taking away write access and granting only read and execute access to owner, group, and other. If you specified 111, you would be taking away execute access and granting read and write access to owner, group, and other. The default umask is 022.

Common values for umask are:

`002` Removes write access for others. Provides read, write, and execute access to owner and group; read and execute access to other.

`022` Removes write access for group and others. Provides read, write, and execute access to owner; read and execute access to group and other.

`027` Removes write access for group and read, write, and execute access for others. Provides read, write, and execute access to user; read and execute access to group; no access to other.

To determine the current value of umask for yourself, execute the `umask` command without arguments:

```
% umask
```

Changing access to files using chmod

Once a file or directory is created, you can change its access permissions using the `chmod` command. `chmod` allows you to specify file-access permissions symbolically or by using file-protection bits expressed in octal digits.

Using the symbolic method, file-access levels and user classes are described by an alphabetic character. The symbolic assignments for user classes are:

`u` = user,

`g` = group,

`o` = other,

`a` = all (user, group, and other)

The symbolic assignments for file access levels are:

`r` = read

`w` = write

`x` = execute

To add the access, precede the file access level with a plus (+) character; to subtract the access, precede the file access level with a minus (-) character. For example, the following command adds read and write file access to everyone for the file named myfile:

```
% chmod a+rw myfile
```

You can use an equal sign (=) to set the access to a specified value and clear out all other accesses for the user class specified. For example, the following command sets the access for group to read and write:

```
% chmod g=rw myfile
```

Using file-protection bits, an octal digit is assigned to each access level. The octal assignments are:

`4` = read access

`2` = write access

`1` = execute access

The sum of the bits describes the set of access levels. For example, if you specify octal 7, you are specifying read (4), write (2), and execute access (1).

A set of three octal digits describes the file-access levels for each file or directory. The first number describes the owner's access, the second number describes the group's access, and the third number describes all other's access. For example, the following command provides read and write access for owner, group, and other for the file named myfile:

```
% chmod 666 myfile
```

You could achieve the same results with the following command:

```
% chmod a=rw myfile
```

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

Public directories

A public directory is a directory that is accessible to all users. Public directories typically hold transient user and system files and are potential avenues for security breaches.

In ConvexOS, public directories are set with mode 0777, granting read, write, and execute privileges to owner, group, and others. This allows any user to manipulate the files contained in them.

Even though a file cannot be read or written by other users, the mode of a public directory allows anyone to remove the files in them regardless of the owner and mode of the file. To prevent this from happening, the superuser or owner can set the sticky bit on directories. Any directory with the sticky bit set restricts removal of a file to the owner of the file and the superuser. Users are denied the right to remove any files except their own.

Only the superuser or the owner of the directory can place the sticky bit on a directory. The sticky bit remains until the owner or superuser either explicitly removes the directory or changes its mode. For more information on the sticky bit, see the sticky(8) man page.

The example in Figure 4 illustrates sticky-bit protection. In this example, the sticky bit is set on the /tmp public directory. User smk in group doc is attempting to manipulate the files in /tmp.

The first command displays the contents of the /tmp directory, their permissions, and their owners.

The second command attempts to remove the file named Ex16566. This action is denied because it is owned by daa, even though it belongs to group doc.

The third command attempts to remove the file named mytemp. This action is permitted as the file is owned by smk.

The `cat` command shows the contents of the file named openfile. This file is owned by root, and grants read and write permissions to everyone.

The first `cp` command copies /dev/null, an empty file, to the file named openfile. Because write permissions are granted to everyone, this action is permitted. The next `cat` command shows that this file is now empty.

The last command in Figure 4 attempts to write /dev/null to the file named protfile. This action is not permitted, because read and write permissions are only granted to the owner of the file.

Figure 4 Sticky-bit protection example

```
% ls -lg /tmp
-rw-rw----    1    daa    doc    19456    Mar 18 21:18 Ex16566
-rw-------    1    smk    doc    279      Mar 17 19:41 mytemp
-rw-rw-rw-    1    root   sys    35       Mar 16 12:27 openfile
-rw-------    1    root   bin    32       Mar 10 10:26 protfile

% rm /tmp/Ex16566
rm: /tmp/Ex16566 not removed.

% rm /tmp/mytemp

% ls -lg /tmp
-rw-------    1    daa    doc    19456    Mar 18 21:18 Ex16566
-rw-rw-rw-    1    root   sys    35       Mar 16 12:27 openfile
-rw-------    1    root   bin    32       Mar 10 1-:26 protfile

% cat /tmp/openfile
You can't remove me.

% cp /dev/null /tmp/openfile
% cat /tmp/openfile

% cp /dev/null /tmp/protfile
cp: /tmp/protfile: Permission denied
```

Clearing suid, sgid bits on write

If a user needs capability for a program not available with their user login ID, they can change their user or group ID to a user or group that has the required capability. Programs run this way are called set-user-ID (suid) and set-group-ID (sgid) programs. These programs have a bit set in their file permissions list to indicate they are suid, sgid, or both. This shows up as an `s` in the execute field in the permissions list.

ConvexOS clears the suid and sgid bits when a file is written to prevent program replacement in an suid or sgid program, unless you are running as root. This way, someone cannot overwrite a program with suid or sgid bits set and inherit their capabilities.

In Figure 5, bit-clearing is demonstrated twice.

Figure 5 Clearing suid/sgid bits

```
% ls -lg myprogram
-rwsrwsrw    1    smith  bin    10240    Jan 11 22:45 myprogram

% cat sneakyprog > myprogram
% ls -lg myprogram
-rwxrwxrwx   1    smith  bin    10240    Mar 18 14:18 myprogram

% ls -lg anotherprog
-rws-------  1    smk    doc    83706    Dec 15 1987 anotherprog

% strip anotherprog
% ls -lg anotherprog
-rwx------   1    smk    doc    17500    Mar 18 14:19 anotherprog
```

The first case demonstrates that bit-clearing occurs when writing files owned by another user. Note the suid and sgid bits are set on the file named myprogram, owned by smith. When user smk overwrites the file myprogram with the file sneakyprog, the suid and sgid bits are cleared.

The second case demonstrates that bit clearing is also performed on files owned by the user initiating the operation. Note the suid bit is set on the file named anotherprog, owned by smk. When user smk strips the file named anotherprog, which rewrites the file, the suid bit is cleared.

Note that the clearing occurs when files are replaced. Adjust any local installation scripts to reset the proper modes.

Preventing misuse of the system

To prevent misuse of the system, consistently use the file-protection methods discussed in this chapter. To protect files:

- Add a `umask` command to the user's .cshrc file to restrict directory access to a group.
- Use the `chmod` command to restrict access to existing files and `umask` to restrict access to files to be created.
- Restrict access to the superuser password.
- Extend write access for password and group files only to superuser.
- Do not extend write privileges to users for system directories, including /, the root directory (mode should be 755). The exception to this is the /tmp directory.
- The /tmp and /usr/tmp directories should allow read, write, and execute access to all users and groups and should have the sticky bit set. See the sticky(8) man page for more details on the sticky bit.

Keep in mind that different types of files require different file access permissions. For example:

- System utility files must be executable; permission to execute these files must be extended to anyone needing to use them.
- Text files are typically created without execute access since they are usually not executable. However, if the text file is a shell script, make the file both readable and executable by changing its file-access permissions using the `chmod` command.
- The loader (`ld`), which links object files and libraries, uses the umask of the user running it. Depending on the value of that user's umask, all users can potentially execute the created files. See the "Default file access" section in this chapter for more details on umask.

Protecting file contents

You can protect the contents of files by erasing deleted files and encrypting existing files.

Erasing deleted files

The ability to "erase" deleted files is another security feature. When enabled, the file-erasing facility overwrites all disk blocks of a deleted file with a value specified by the system manager; blocks from deleted files retrieved directly from the raw disk are not readable. Enabling this utility increases system overhead and degrades performance.

To enable file erasing, set the kernel boot-time parameter option `erase_unlink` to 1. You can use the `erase_pattern` kernel boot-time parameter to specify the 32-bit pattern to use to overwrite deleted files. Refer to Chapter 15, "Customizing kernel boot-time parameters," on page 287, for details on setting these parameters.

When creating a new file system using either the `newfs` or `newst` utility, you can also initialize a file system with an erase pattern by placing the `-E` pattern argument as the first argument to the command. This overwrites the partition with the pattern before creating the file system.

Encrypting files

Do not put sensitive material such as payroll data, confidential memos, strategic information, or classified data online without encrypting it. The `crypt` utility encrypts and decrypts the contents of a file based on a password; the password is the key that selects a particular transformation for encryption. (Use the `ccrypt` utility for international releases of ConvexOS.)

The format for encrypting a file is

`crypt` < *plain_file* > *encrypted_file* [*key*]

where

plain_file	is the source file to be encrypted.
encrypted_file	is the output encrypted file.
key	is the key by which the file is encrypted. If you do not specify a key, the `crypt` utility prompts for a password and inhibits echoing to the terminal while the password is entered.

The following example illustrates the `crypt` command,

```
% crypt < test.data > test.hidden
```

and the key prompt:

```
Enter key: $
```

Enter the key by which you want the file encrypted. When the file is encrypted, remove the original (plain text) file from the system. For example, enter

```
% rm test.data
```

The format for decrypting a file is:

`crypt` < *encrypted_file* > *plain_file* [*key*]

You must use the same key used when encrypting the file. The encryption algorithm is difficult, but not impossible, to break. Because the key and key security are the most vulnerable aspect of `crypt`, make the key at least six letters and do not store it on the system.

See the crypt(1) man page for more details on encrypting and decrypting files.

Protecting mail files

ConvexOS automatically creates and maintains mail files. The system creates a mailbox when the first piece of mail is sent to a user. When a reader deletes all the mail, /usr/ucb/mail deletes the mailbox file. Refer to the mail(1) man page for mail system documentation.

The following methods are used by the mail programs to secure the mail system:

- Access permissions are set on the mail directories so that only the superuser is allowed to write files to the mail directory.
- Owner access permissions are set on mail files so only the owner of the mail is allowed to read it.
- Individual pieces of mail in transit are created as root-owned files in the directory /usr/spool/mqueue and read-protected from users. File ownership transfers to the recipient when the mail is delivered and placed into the directory /usr/spool/mail.

Protecting the system using log files

ConvexOS provides log files that help secure the system by logging failed login attempts and failed attempts to access files. Use the information in these log files to detect attempts to breach security. Refer to Chapter 12, "Setting up log files," on page 251, for details on setting up these log files.

Logging failed file-access attempts

For additional system security, ConvexOS provides a facility for logging file-access attempts that fail because of insufficient file-access permissions. With this facility, you can track unauthorized attempts to access protected files or directories. Enabling this utility increases system overhead and degrades system performance.

The /usr/adm/failure_log file contains an entry for each file access attempt that fails because of insufficient permissions. It contains enough information to determine who attempted the access, what command was used, the date and time the attempt was made, and the file involved.

System calls in the following list can generate log messages when they fail.

- access
- acct
- bind
- chdir
- chmod
- chown
- connect
- creat
- execve
- exportfs
- faillog
- link
- lstat
- open
- mknod
- mkdir
- mount
- quotacl
- relink
- rename
- stat
- statfs
- swapon
- symlink
- truncate
- unlink
- unmount
- utime

A failed attempt to generate a core file also generates a log entry.

Logging failed login attempts

You can further protect your system by logging failed login attempts to a user-named file. You specify this in the syslog.conf file. Refer to Chapter 12, "Setting up log files," on page 251, for details on how to do this.

Adding devices

2

This chapter describes how to reconfigure ConvexOS to add new physical devices and pseudodevices. The following topics are included:

- The /ioconfig file located on the service processor unit (SPU)
- Supported controllers and devices
- Special device files
- Adding a disk
- Adding terminals
- Configuring pseudoterminals
- Adding a modem
- Adding printers and plotters

Adding CONVEX supported devices is primarily a hardware task. Refer to the documentation for the hardware you are adding as well as to the *CONVEX Guide to Attaching Multibus Peripherals*, the *CONVEX VMEbus Service Kit* if you are adding VME peripherals, the *CONVEX HSP User's Guide* if you are adding a High Speed Parallel (HSP) Interface, and the *CONVEX Integrated Disk Channel Service Guide* if you are adding an IDC.

If you are adding new disk drives, refer to Chapter 3, "Setting up the disk system," for information about configuring disks, creating partitions, and setting up file systems.

If you are adding a device that is not supported by CONVEX, refer to the *CONVEX Guide to Writing Device Drivers*. This guide is part of the optional User-Written Device Drivers package and includes instructions on writing and installing device drivers and generating a new operating system.

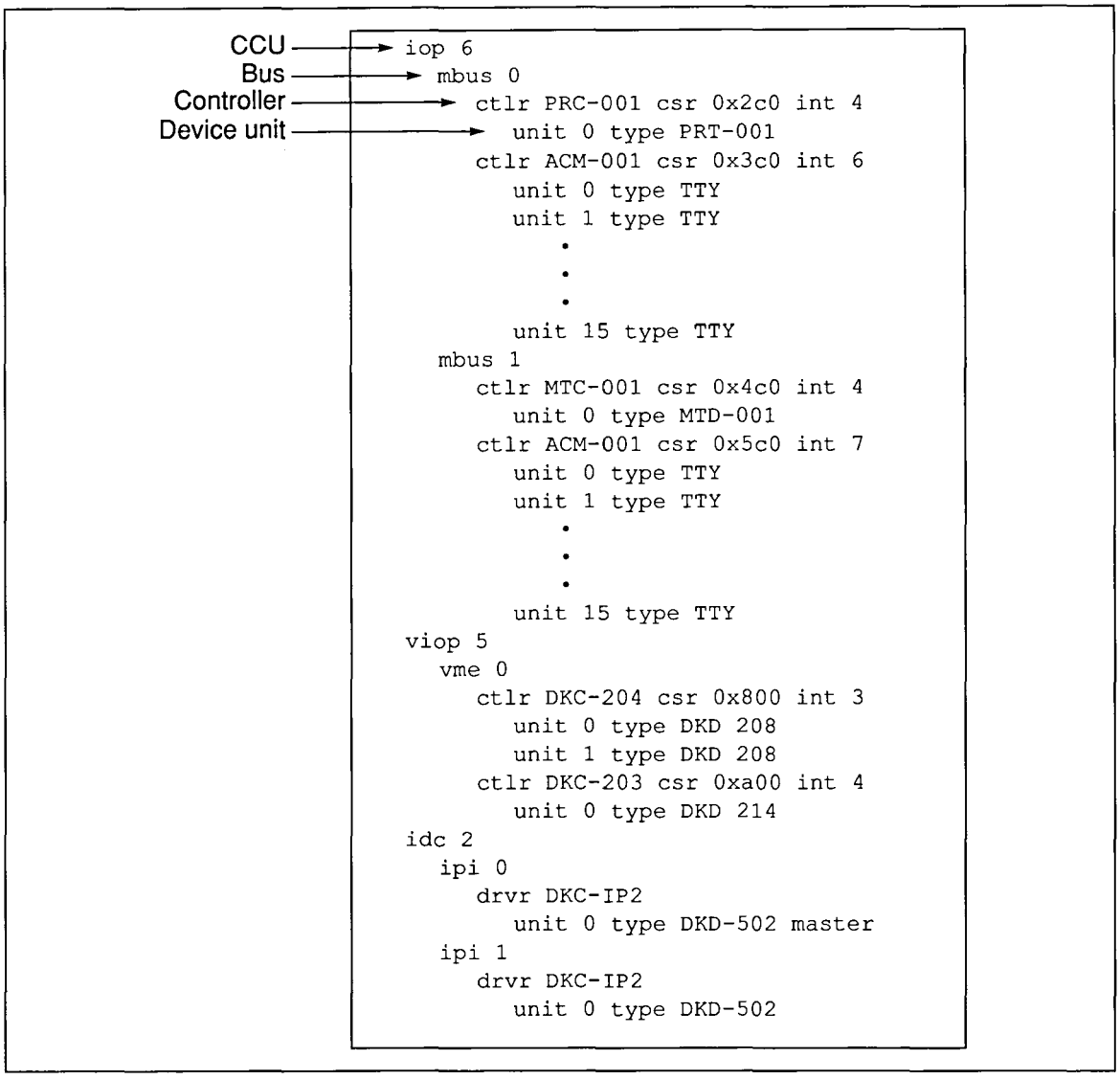
The /ioconfig file

When adding a device to your system, you must integrate the new device into ConvexOS by modifying the /ioconfig file. The /ioconfig file contains a description of all channel control units (CCUs), interfaces, controller boards, and peripheral devices for your system. The boot process reads this file to determine what devices are present.

The /ioconfig file is located on the SPU disk. A sample /ioconfig file is shown in Figure 6. To display this file, enter:

```
% spu -r /ioconfig | more
```

Figure 6 Example /ioconfig file



The /ioconfig file contains an entry for each CCU connected to the system. Each entry contains several levels of information. These levels are usually distinguished by indentation for readability, although indentation is not required. The following sections describe the different levels of information in the /ioconfig file.

For more information, refer also to Appendix C, "Controller, device, and driver /ioconfig designations," on page 403.

CCU description

Beginning at the left margin, the CCU description includes the CCU type and slot number. Supported CCU types include:

- `iop` (Multibus Input/Output Processor)
- `viop` (VMEbus Input/Output Processor)
- `hsp` (High Speed Parallel Channel Controller)
- `idc` (Integrated Disk Channel Controller)
- `tli` (Tape Library Interface)
- `hippi` (High Performance Parallel Interface)

Slot numbers range from 0 to 14, corresponding to the I/O slot to which the CCU is connected. The number of CCUs supported by your system depends on the CONVEX processor series. This can be one of the following:

- C100 Series models support 5 CCUs in slots 3 through 7. Note that TLI and IDC CCUs are not supported in C100 Series machines.
- C200 Series models C201, C202, C210, C220, and model C240 with 1 PBUS, support 4 CCUs in slots 0 through 3.
- C230 and C240 models with 2 PBUSs support 8 CCUs in slots 0 through 7.
- C230I model with 4 PBUSs supports 14 CCUs in slots 0 through 7, slots 8 through 10, and slots 12 through 14.
- C3400 Series models with 1 PBUS support 4 CCUs in slots 0 through 3.
- C3400 Series models with 2 PBUSs support 8 CCUs in slots 0 through 7.
- C3800 Series systems can have up to 24 CCUs.

C3800 Series systems can have up to four CPU bays (bays 0 through 3) and one I/O bay (bay 4). Both CPU and I/O bays have eight slots per bay, four on each side. At least one side of each CPU bay must contain CPU boards, the other side can contain up to four CCUs. The I/O bay can contain eight CCUs.

Table 1 shows the possible slot numbers for CCUs in a fully configured C3800 Series system.

Table 1 CCU slot number for C3800 Series system

Bay	Slot numbers
0 (CPU)	0-3 or 4-7
1 (CPU)	8-11 or 12-15
2 (CPU)	16-19 or 20-23
3 (CPU)	24-27 or 28-31
4 (I/O)	32-39

Bus or interface description

Information at the first level of indentation includes the type and chassis number of the bus or interface. Supported types are:

- Multibus (`mbus`) for IOP-type CCUs. IOP-type CCUs support two Multibuses.
- VMEbus (`vme`) for VIOP-type CCUs. VIOP-type CCUs support two VMEbuses.
- Intelligent Peripheral Interface (IPI) IDC-type CCUs. IDC-type CCUs support 4 IPIs.
- HSP-type CCUs do not have controller chassis.

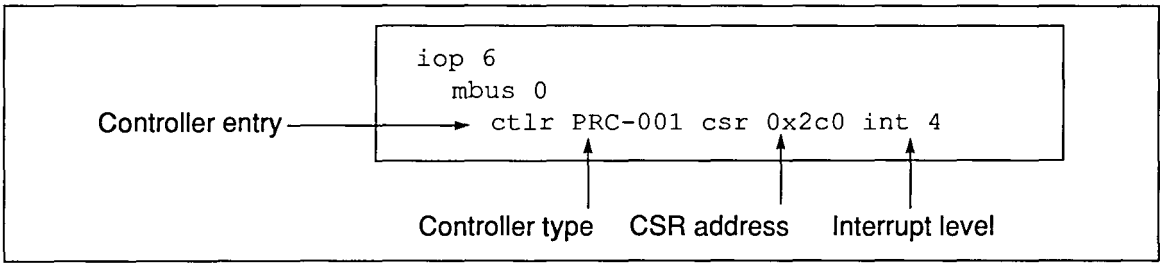
Numbering begins with zero (0) for the first bus or interface on each CCU and increments sequentially for each additional bus or interface on the same CCU.

Controller or driver description

At the second level of indentation, the controller description specifies the I/O controller type or user-supplied driver type, depending on the type of CCU.

For IOP- and VIOP-type CCUs, the `ctlr` line entry specifies the control and status register (CSR) address and interrupt levels. Figure 7 illustrates an I/O controller entry in the /ioconfig file.

Figure 7 IOP and VIOP controller unit entry in /ioconfig file



The format for an I/O controller description is

`ctlr` *type* `csr` *address* `int` *level*

which has the following components:

type Type of controller.

address Control and status register address of the controller

level Interrupt level for the controller. The interrupt level can be any integer between 0 and 7.

Caution

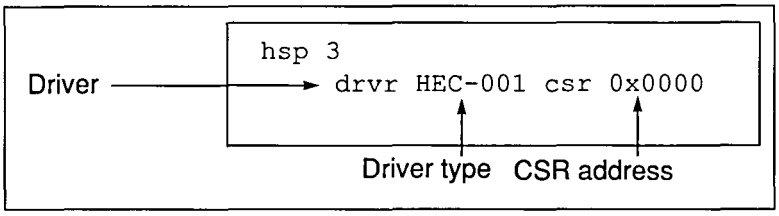
The CSR and interrupt parameters are configured by CONVEX field support specialists and are site-specific. Contact the CONVEX Technical Assistance Center (TAC) before modifying these parameters.

The CSR address and interrupt levels for a controller are provided in the specific controller documentation and in the *CONVEX Guide to Attaching Multibus Peripherals* and *CONVEX VMEbus Service Kit*.

HSP driver

For HSP-type CCUs, the second level of indentation specifies the driver type and CSR address. HSP drivers do not have an associated interrupt level. Figure 8 illustrates an HSP driver entry in the /ioconfig file.

Figure 8 HSP driver entry in /ioconfig file



The format for an HSP driver description is

`drvr` *type* `csr` *address*

where

type is the type of driver.

address is the control store register address of the driver.

Caution

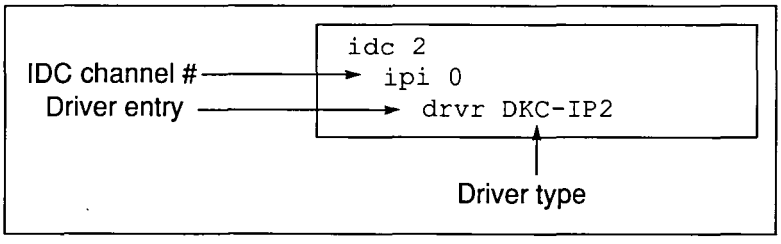
The CSR address is configured by CONVEX field support specialists and is site-specific. Contact the CONVEX Technical Assistance Center (TAC) before modifying these parameters.

The CSR address for a controller is provided in the specific controller documentation and in the *CONVEX Guide to Attaching Multibus Peripherals* and *CONVEX VMEbus Service Kit*.

IDC drivers

IDC-type drivers do not have an associated CSR address or interrupt level. Figure 9 illustrates an IDC driver entry in the /ioconfig file.

Figure 9 IDC driver entry in /ioconfig file



The format for an IDC driver description is

`drvr` *type*

where *type* is the type of driver.

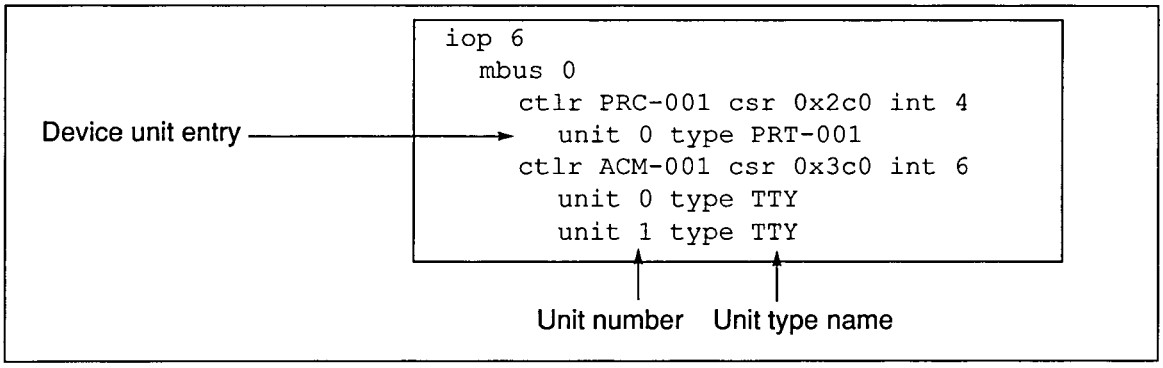
Device unit

At the third level of indentation, the device unit description specifies the unit number and type name for I/O or channel units, depending on the type of CCU to which it is connected.

IOP, VIOP, and IDC controllers

Figure 10 illustrates an entry in the /ioconfig file for a device unit attached to an IOP-type controller.

Figure 10 Device unit entry in /ioconfig file for IOP controllers



For IOP-, VIOP-, and IDC-type controllers, the format for the unit description is

```
unit number type name [master]
```

where

number is the number assigned to the unit. Units are numbered sequentially beginning with 0 for each controller type on a bus. The number of units supported by a single controller depends upon the controller type. You can find this information in the documentation for the specific controller.

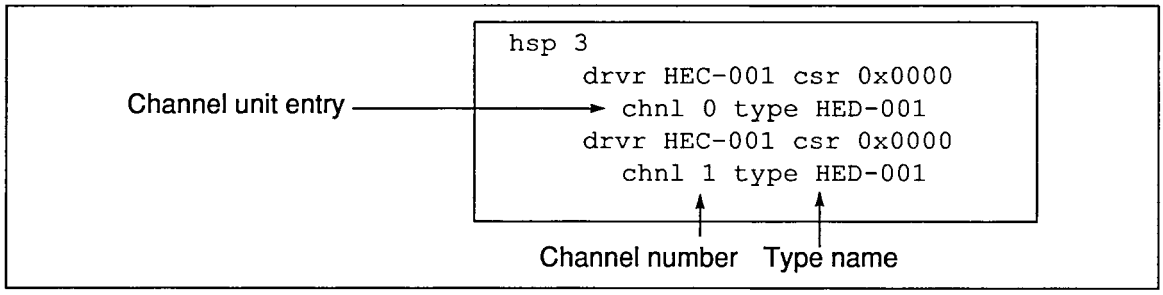
name is the type of unit connected to the controller.

`master` is a keyword that applies only to IDC-type CCUs. For a discussion of this keyword, refer to the *CONVEX Integrated Disk Channel Service Guide.*

HSP CCU

HSP-type CCUs have associated channels rather than units. Figure 11 illustrates an entry in the /ioconfig file for a device unit attached to an HSP-type controller.

Figure 11 Device unit entry in /ioconfig file for HSP controllers



The format for specifying HSP channels is

`chnl` *number* `type` *name*

which consists of the following components:

number Channel number. Channels are numbered sequentially beginning with 0 for each HSP on a bus.

name Type of channel connected to the HSP

Device files

Physical devices are accessed by users through device files. A device file is a special file that represents an I/O device. Physical devices are accessed through these special files in the same way an ordinary file is accessed. Each I/O device connected to the system has at least one special device file associated with it.

Special device files are created using the `mknod` command. For convenience, a shell script called `MAKEDEV` is provided in the /dev directory that allows you to supply default values to `mknod`, reducing the number of steps required to create device files.

Caution

Executing `MAKEDEV` will reset ownership and modes of device files to default values. CONVEX recommends that you read and understand the `MAKEDEV` script before executing it.

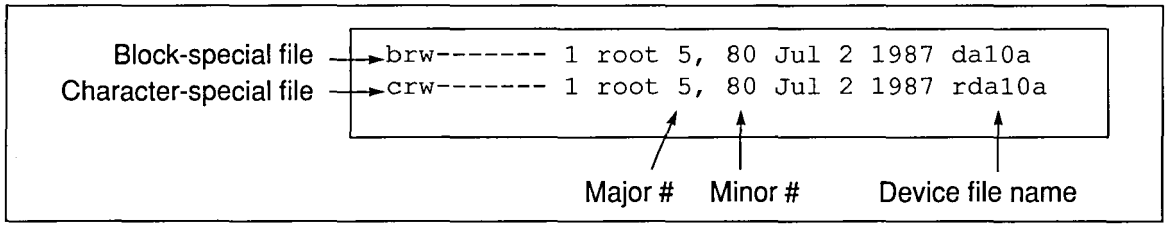
CONVEX ships files for a standard system; you do not have to execute `MAKEDEV` unless you add devices. Refer to the makedev(8) man page for more information.

Device files are kept in the /dev directory. There are two types of device files: block special device files and character special device files:

- Block special device files access buffered devices that transfer information in blocks, such as disk and tape devices.
- Character special device files are used to access devices that typically transmit only one character or line at a time, such as terminals and printers.

Block devices, however, can also be accessed in character mode. Consequently, for each block special device file created, a corresponding character special device file is created. This corresponding device is sometimes referred to as the raw device. Raw device file names begin with an `r`; block device names omit the *r*. An example of a block and character special device file entry in /dev is shown in Figure 12.

Figure 12 Example special device file entries in /dev



Each device file has a major and minor number that you assign when you create the device file. The /dev/MAKEDEV shell script automatically assigns the appropriate major and minor numbers for supported devices. For unsupported devices, you must edit the MAKEDEV script to include the new device type, or create the device files manually using the `mknod` command. The major number identifies which device driver communicates with the device, thus indicating what type of device it is (disk, tape, printer, etc.). The minor number indicates the actual device (such as disk 1 on controller 0). In some cases, the minor number specifies particular characteristics of the device. For example, a single tape drive has several device files; each device file has a unique minor number that represents various configurations, such as recording density and whether or not to rewind.

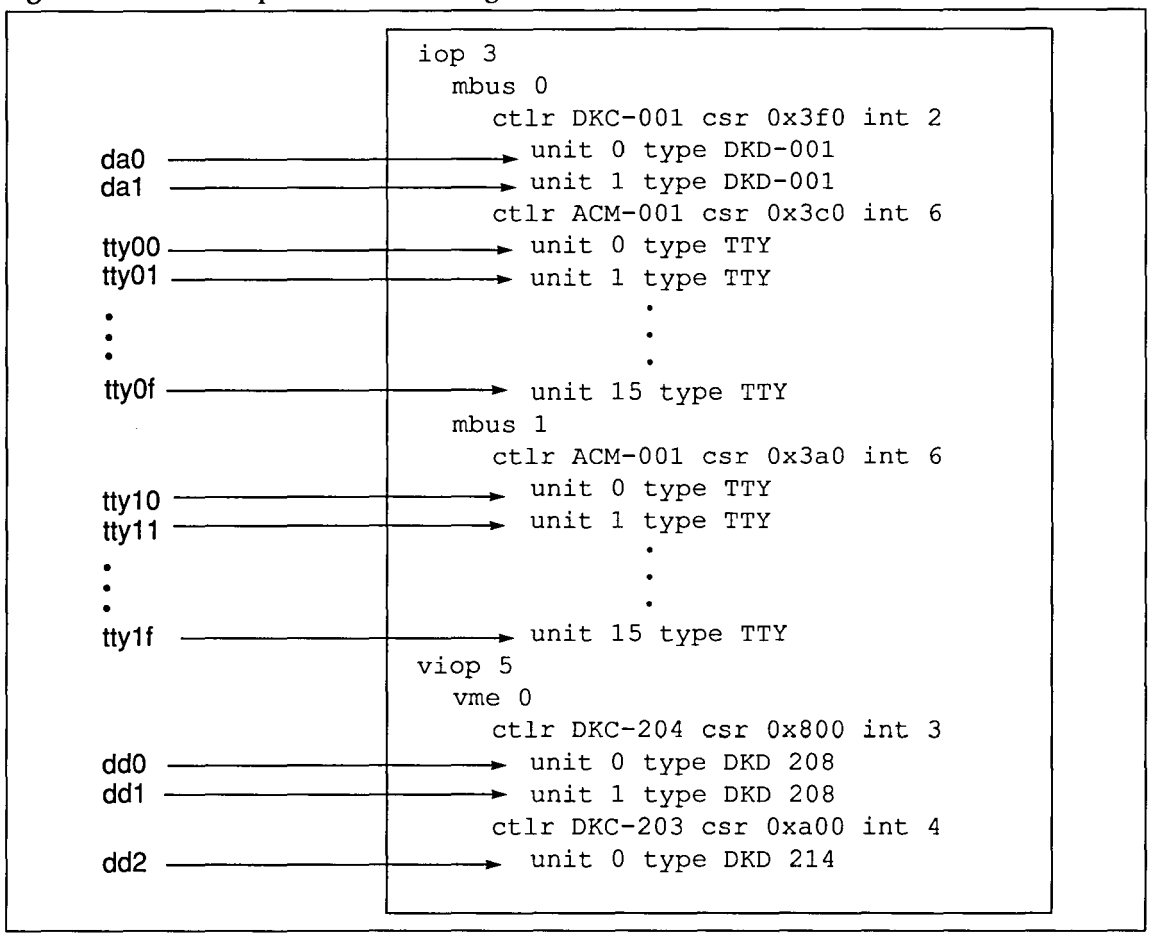
Device files are named according to device type and their placement in the /ioconfig file. Table 2 lists the device types available on a CONVEX machine and the code used in the device file name for each device type.

Table 2 ConvexOS device file naming conventions

Device type	Name
9-track reel tape	mt*n* (block)
	rmt*n* (raw)
3480 cartridge tape	tc*n* (block)
	rtc*n* (raw)
DAT tape	dat*n* (block)
	rdat*n* (raw)
Disk stripes	st*n* (block)
	rst*n* (raw)
DR11-W interface	dm*n*
Ethernet	ex*n*
HYPERchannel	hy*n*
IDC disk	du*n*[a-h] (block)
	du*n*[a-h] (raw)
Labeled tape	lt*n* (lt/c*n* and lt/u*n*)
Line printer	lp*n*
Multibus disk	da*n*[a-h] (block)
	rda*n*[a-h] (raw)
Plotter	pb*n*
Pseudoterminal	pty[e-t]*n*
RAM disk	ramd*n* (block)
	ramc*n* (raw)
Terminal controller	tty*n*
VMEbus disk	dd*n*[a-h] (block)
	rdd*n*[a-h] (raw)

Device units are numbered sequentially, starting with 0 for each controller type, according to their order in the /ioconfig file. For example, the first Multibus disk listed would be named da0, the next da1, and so forth. The first VMEbus disk listed would be named dd0, the next would be dd1, and so forth. All VME and Multibus devices should be placed first (before IDC or TLI) in the ioconfig file. Figure 13 shows the relationship between entries in /ioconfig and special device file names in the /dev directory.

Figure 13 Relationship between /ioconfig and device files



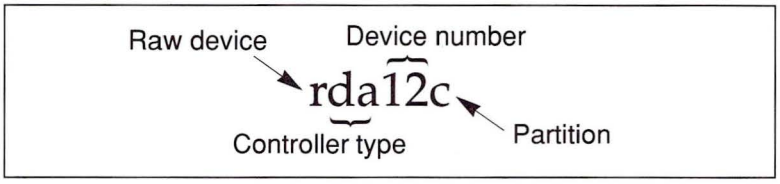
Naming convention for disk devices

For disk devices, the device file name indicates the following:

- Whether the named device is a character special file (raw device) or block special file. Raw disk device names begin with an r; block device names omit the r.
- The type of controller the device is connected to (mbus=da, vme=dd, ipi=du).
- The unique number assigned to the device.
- The partition name. Because disks are partitioned, multiple device files are created for each disk (one block access and one character access file for each partition `a` through `h`, for a total of 16 device files per disk). Refer to Chapter 3,"Setting up the disk system," for information about creating partitions.

Figure 14 explains the components of a disk device name.

Figure 14 Disk device name fields



Adding a disk

Complete the following steps to add a new disk to the system configuration:

Step 1 Install the physical hardware. If this is an unsupported device, you must install a device driver. Refer to the *CONVEX Guide to Writing Device Drivers* for information about writing and installing drivers.

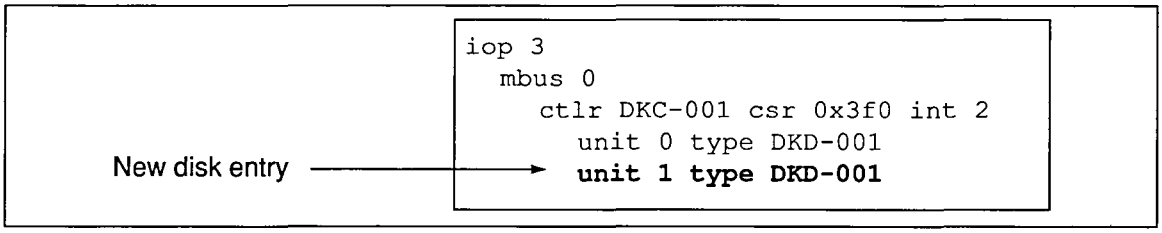
Step 2 Power up the system and boot to the SPU level. Information on powering up and booting the system can be found in *CONVEX SPU System Manager's Guide* or the *CONVEX C3800 Series SPU System Manager's Guide,* whichever is available at your site.

Step 3 Make a back-up copy of the /ioconfig file on the SPU. Enter:

```
(spu)> cp /ioconfig /ioconfig.old
```

Step 4 Edit the /ioconfig file to include information about your device. The amount of information you add depends on the hardware you installed. Figure 15 illustrates adding a second disk device to an existing controller.

Figure 15 Adding a second disk device to an existing controller



Refer to the section, "The /ioconfig file," on page 22, for more information on the structure of this file.

Step 5 Change to the /mnt/os directory. Enter:

```
(spu)> cd /mnt/os
```

Step 6 Boot the system by entering:

```
(spu)> boot
```

Step 7 Check to make sure that the device was found during the boot process.

Step 8 Log in as the superuser.

Step 9 If this is a new disk type not supported by CONVEX, you must edit the /etc/disktab file to include a description of the new disk. An example /etc/disktab entry is shown in Figure 16.

Figure 16 Example /etc/disktab file

```
dkd-001|dkd001|DKD-001|DKD001|eagle|Eagle|Fujitsu Eagle (45 sectors):\
      :ty=winchester:se#512:ns#45:nt#20:nc#842:rm#3900\
      :pa#37800:ba#8192:fa#1024:\
      :pb#151200:bb#8192:fb#1024:
      :pc#756000:bc#65536:fc#8192:\
      :pd#37800:bd#8192:fd#1024:\
      :pe#227700:be#8192:fe#1024:\
      :pf#75600:bf#8192:ff#1024:\
      :pg#341100:bg#8192:fg#1024:\
      :ph#225900:bh#8192:fh#1024:
```

The /etc/disktab file is a database that describes disk geometries and disk partition characteristics. The codes used in this file are:

`ty`	Specifies type of disk.
`se`	Specifies sector size in bytes.
`ns`	Specifies sectors per track.
`nt`	Specifies tracks per cylinder.
`nc`	Specifies total cylinder count on the disk.
`rm`	Specifies speed in revolutions per minute.
`b[a-h]`	Specifies partition block size.
`f[a-h]`	Specifies partition fragment size.
`p[a-h]`	Specifies size of partition.

Refer to the disktab(5) man page for more information.

Caution

Changing the values of standard disks in /etc/disktab will not change the values used by device drivers. Call the CONVEX TAC before you attempt to modify values in this file.

Step 10 Partition the disk and create block and character special device files using MAKEDEV. Refer to Chapter 3, "Setting up the disk system," on page 65, for information on how to do this.

Adding terminals

Complete the following steps to configure additional terminals. ConvexOS supports up to 256 teletype (tty) lines, depending on licensing agreements.

Step 1 Install the physical hardware. If this is an unsupported device, you must install a device driver. Refer to the *CONVEX Guide to Writing Device Drivers* for information on writing and installing drivers.

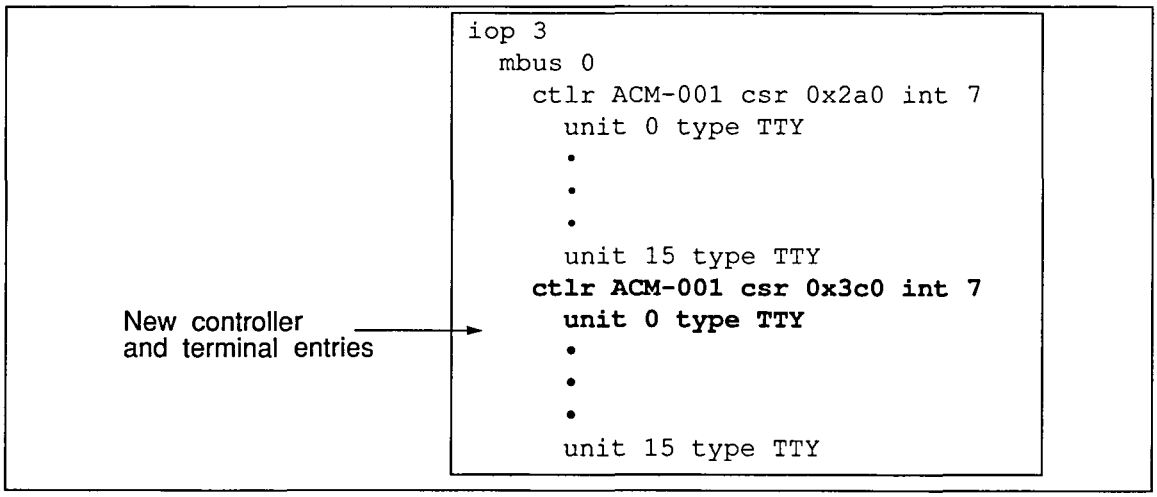
Step 2 Power up the system and boot to the SPU level. Information on powering up and booting the system can be found in *CONVEX SPU System Manager's Guide* or the *CONVEX C3800 Series SPU System Manager's Guide.*

Step 3 Make a backup copy of the /ioconfig file on the SPU. Enter:

```
(spu)> cp /ioconfig /ioconfig.old
```

Step 4 Edit the /ioconfig file to include information about your device. The amount of information you add depends on the hardware you installed. Figure 17 illustrates adding an additional asynchronous communications controller and 16 tty devices to an existing Multibus CCU.

Figure 17 Adding an additional asynchronous communications controller



Refer to the section, "The /ioconfig file," on page 22, for more information on the structure of this file.

Step 5 Change to the /mnt/os directory. Enter:

```
(spu)> cd /mnt/os
```

Step 6 Boot the system by entering:

```
(spu)> boot
```

Step 7 Check to make sure that the device was found during the boot process.

Step 8 Log in as the superuser.

Step 9 If necessary, create special device files for the device using the MAKEDEV script. For example, to create device entries for a second terminal controller, enter:

```
# cd /dev
# MAKEDEV ca1
```

MAKEDEV creates the files tty10 through tty1f.

Note

The order of terminal controllers listed in the /ioconfig file determines the numbering of controller names and the set of /dev/tty*xx* files used by that controller.

ConvexOS uses tty [0-9,A-F][0-f] for terminal names, as shown in Table 3. The lines on the first ca controller are named /dev/tty00, /dev/tty01,..., /dev/tty0f. If a configuration has more than one ca interface, successive 16-line terminal groups are named as shown in Table 3. The names tty [G-Z][0-f] are reserved by ConvexOS/Secure.

Table 3 Terminal naming conventions

Controller	Terminals
ca0	tty00 through tty0f
ca1	tty10 through tty1f
⋮	⋮
ca9	tty90 through tty9f
ca10	ttyA0 through ttyAf
⋮	⋮
ca15	ttyF0 through ttyFf

Step 10 Edit the /etc/ttys file to add entries for each tty device created with MAKEDEV. The init program uses the ttys file to determine whether or not getty should be run on the terminal line.

Because CONVEX ships this file with entries for many devices already in place, editing this file is necessary only if you add more terminals than are listed in /etc/ttys. A portion of the /etc/ttys file shipped with ConvexOS is shown in Figure 18.

Figure 18 Example /etc/ttys file

```
console"/etc/getty std.9600"   vt100n   on    secure
tty00  "/etc/getty std.9600"   vt100n   on
tty01  "/etc/getty std.9600"   vt100n   on    dialup
tty02  "/etc/getty std.9600"   vt100n   on    dialup   uucp
tty03  "/etc/getty std.9600"   vt100n   on             <engr>
tty04  "/etc/getty std.9600"   vt100n   on
```

The format of the /etc/ttys file is shown below with a description of each keyword. Refer to the ttys(5) man page for more information.

name command type `[on|off] [secure] [dialup] [uucp]` [*access*]

where

name is the terminal name as listed in /dev.

command is the command to execute after initialization, usually `getty`. This field also specifies the name of the /etc/gettytab entry that contains the terminal line definitions for this tty line, for example, std.9600.

type is the terminal type as listed in the /etc/termcap file.

`on/off` specifies whether or not the `init` process executes the command specified in *command*. Listing a line as `off` disallows logins on that line.

`secure` specifies that root can log in on this port. When specified, the line is assumed to be "secure;" root can only log in on lines marked `secure`. Note, however, that this does not affect the use of the `su` command. CONVEX recommends that only the console line be marked `secure`.

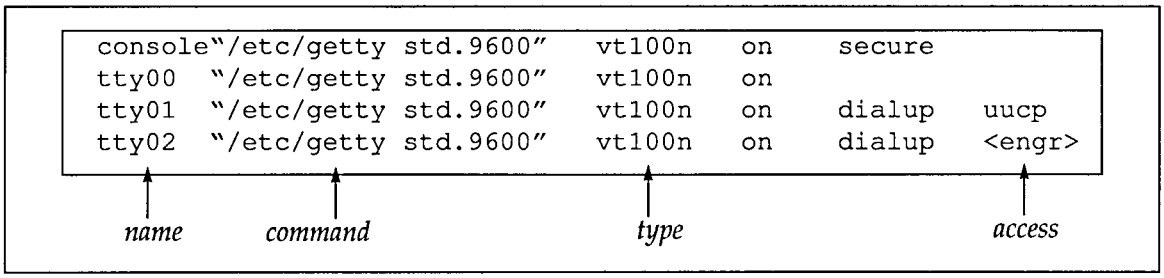
`dialup` specifies whether users are required to enter a dial-in password. This is usually used when logging in over telephone lines. If blank, no password is required.

`uucp` allows users belonging to group uucp (group 40) to log in.

access specifies which users are allowed to log in. You can selectively allow or disallow logins by users or groups by specifying an access control list. Users or groups whose names appear in the list are allowed access; preceding a name with an exclamation point (!) denies access to the user or group. Group names are placed between less-than (<) and greater-than (>) symbols.

Entries in the access control list are read from left to right. Be careful of the order in which you list users or groups. If you explicitly give a user or group access, all users whom you want to have access must be specified. If there is only a denial list, a user or group can log in as long as that user or group is not one of those explicitly denied access. Figure 19 shows an example of this.

Figure 19 Example /etc/ttys file with user access specified



Step 11 If you specified a terminal line definition in Step 10 for an entry that does not currently exist in the /etc/gettytab file, edit this file to include the characteristics of the new device.

The /etc/gettytab file contains terminal line definitions. Each time `getty` starts, the `getty` process uses this file to determine a tty line's characteristics. Figure 20 illustrates a sample /etc/gettytab file.

Figure 20 Sample /etc/gettytab file

```
default:\
  :ap:fd#1000:\
  :im=\r\nConvex Systems (%h)\n\r:\
  :sp#1200:
2|std.9600|9600-baud
  :sp#9600:
d1200|Dial-1200:\
  :nx=d150:fd#1:sp#1200:
```

The format of this file is:

name | *alternate_name* [| ...] :
:*attribute*:*attribute*:

where

name	is a one-character entry name.
alternate_name	is an optional alternate name or list of names. Any of the names specified can be used in the *command* field of the /etc/ttys file to reference this entry. The names are separated by vertical bars (|).
attribute	specifies attributes for terminal lines. Some examples are:
`ap`	Defines the parity.
`ht`	Defines hardware tabs.
`ep`	Defines the even parity.
`sp#`*speed*	Defines the line speed.
`im=`*message*	Defines the initial message displayed on the terminal.
`lm=`*prompt*	Defines the login prompt that appears on the terminal.
`nx=`*next*	Identifies the *next* entry to use if `getty` detects a line break.
`cd#`*msec*	Carriage return delay in milliseconds.
`fd#`*msec*	Form-feed delay in milliseconds.

Consult the gettytab(5) man page for more information when modifying this file.

Step 12 If you specified a terminal type in Step 10 that does not exist in the /etc/termcap file, edit this file to include the characteristics of the new device. The /etc/termcap file contains information on terminal capabilities. Figure 21 illustrates a sample /etc/termcap file.

Figure 21 Sample /etc/termcap file

```
d0|vt100n|vt100n terminal:\
        :co#80:li#24:cl=50\E[H\E[2J:\
        :bs:cm=5\E[%i%d;%dH:nd=\E[C:up=\E[A:\
        :al=3\E[L:dl=3\E[M:ku=\E[A:kd=\E[B:kr=\E[C:\
        :kl=\E[D:\do=\E[B:\
        :ic=\E[@:ei=:im=:pt:bw:dc=\E[P:ce=3\E[K:
        :ho=10\E[H:pt:\
        :mi:nd=\E[C:bt=\E[Z:us=\E[8p:\
        :ue=\E[p:so=\E[5m:se=\E[m:\
        :md=\E[1p:mr=\E[16p:mb=\E[2p:mk=\E[4p:\
        :me=\E[0p:hs:ll=\E[24;1H:\
```

The format of this file is

name|*alternate_name*[|...]:
:*attribute*:*attribute*:

where

name	is a one-character entry name.
alternate_name	is an optional alternate name or list of names. Any of the names specified can be used in the *type* field of the /etc/ttys file to reference this entry. Names are separated by pipes (|).
attribute	specifies the attributes for the terminal type. Consult the termcap(5) man page before modifying this file.

Step 13 If you edited the /etc/ttys file in Step 10, reinitialize the file with the on command by entering

```
% on -s
```

Caution

The on command effectively updates all tty ports.

Configuring pseudoterminals

A pseudoterminal is implemented as a pair of character devices. They are used to support remote logins, networking programs, X Windows, /usr/ucb/window, the `emacs` editor, and the `script` utility. Devices are linked in a master/slave relationship; everything written on the master device is input to the slave device and everything written on the slave device is input to the master device. Logins are disabled on pseudoterminals in /etc/ttys. See the pty(4) man page for details.

Complete the following steps to configure a pseudoterminal:

Step 1 Log in as the superuser.

Step 2 If the pseudoterminal device files are not on your system, use the following commands to create the first 16 pairs:

```
# cd /dev
# MAKEDEV pty0
```

MAKEDEV creates the following files in /dev:

- ptyp0 through ptypf
- ttyp0 through ttypf

MAKEDEV creates pseudoterminal pairs in groups of 16. To create additional groups of 16, use MAKEDEV with pty*n* as an argument, where *n* is a number from 0 through 15.

Note

The number of pseudoterminals that ConvexOS supports is defined in the NUMBER_PTYS boot-time parameter.

The naming convention used by MAKEDEV for pseudoterminals is

/dev/pty*XY* (master)

/dev/tty*XY* (slave)

where

X is a one-character group ordinal, e through t.

Y is a one-digit hexadecimal number, 0 through f.

Step 3 Add an entry to /etc/ttys for each pty pair created and ensure they are all listed as `off`.

ConvexOS supports a maximum of 256 pseudoterminal pairs, with a default of 64. `MAKEDEV` creates the pseudoterminals in pairs, with ptyp as the master and ttyp as the slave. When you edit /etc/ttys, add only the slave (ttyp*n*). Do not add the pty half of the pseudoterminal pair. Figure 22 illustrates pseudoterminal entries in the /etc/ttys file.

Figure 22 Example /etc/ttys file showing pseudoterminal entries

```
ttyp0   "/etc/getty std.9600"   vt102    off
ttyp1   "/etc/getty std.9600"   vt102    off    secure
ttyp2   "/etc/getty std.9600"   vt102    off    secure
```

Refer to the "Adding terminals" section in this chapter for more information on editing the /etc/ttys file.

Adding a modem

This section explains how to configure a modem to work with ConvexOS. ConvexOS supports the following modems:

- Trailblazer Plus and Trailblazer Plus/T2000
- Racal-Vadic VA212
- Maxwell 1200VP

CONVEX serial ports are, by default, Data Communications Equipment (DCE) ports intended to be directly connected to Data Terminal Equipment (DTE) devices (terminals). This can be changed, on a port-by-port basis, by installing a modem plug (CONVEX part number 221-000001-203). The modem plug converts a specific port to a DTE device for direct connection to a modem (which should always be a DCE device).

To open a tty port on a CONVEX computer, the RS-232 signals DCD (data carrier detect) and DSR (data set ready) must be asserted. In the case of a terminal, these signals are usually used to indicate that the terminal is powered on. In the case of a modem, there are two possible sets of conditions that determine the state of these signals, because connection to a modem may be initiated from different sources (the local system or a remote modem via the telephone line). Each is described below:

- For dialing out (either `tip` or UUCP), the DCD/DSR signals must be asserted by the modem at all times or ConvexOS will not be able to open the port to talk to the modem. This is best accomplished by settings on the modem that allow the modem to cycle DSR when a disconnect occurs, but assert DCD and DSR at all other times. Refer to your modem manual for information on how to make these settings. If DSR and DCD are hardwired on, the software will not be able to detect a disconnect. In this case, noninteractive software (such as UUCP) will normally timeout; however, interactive software (such as `tip`) will not automatically free the line or device on a remote disconnect.
- If the modem is used as a dial-in line, DSR should follow the off-hook condition of the modem, and DCD should be asserted in the presence of a carrier signal from the remote modem. In this situation, if the remote modem is disconnected for some reason, the local modem will drop DSR and DCD, causing ConvexOS to terminate the current connect session. The `init` process then blocks when reopening the port because DSR and DCD are not asserted; `init` will stay blocked until another modem connects to assert those signals, at which time the `open` will succeed, and `getty` will be forked to print the `login` prompt.

Setting up hardware

Complete the following steps to set up the modem hardware:

Step 1 Connect the modem to an available tty port. If you need to install asynchronous communications controllers to gain additional ports, refer to the "Configuring terminals" section for information on installing these controllers. Figure 23 and Figure 24 illustrate cable wiring pin requirements for connecting modems to a CONVEX system.

Figure 23 Computer-to-modem cable pinout (with modem plug)

CONVEX Pin		Modem Pin	
1	———	1	Frame Ground
2	———	2	Transmit Data
3	———	3	Receive Data
4	———	4	Request to Send
5	———	5	Clear to Send
6	———	6	Data Set Ready
7	———	7	Signal Ground
8	———	8	Data Carrier Detected
20	———	20	Data Terminal Ready

Modem Plug part number: 221-000001-203

Figure 24 Computer-to-modem cable pinout (without modem plug)

CONVEX Pin		Modem Pin	
1	———	1	Frame Ground
2	———	3	Receive Data
3	———	2	Transmit Data
4	———	5	Clear to Send
5	———	4	Request to Send
6	———	20	Data Terminal Ready
7	———	7	Signal Ground
8	———	8	Carrier Detect
20	———	6	Data Set Ready
22	———	22	Ring Indicator

Part number: 604-100001-*xxx*; modem end is male. Part number 604-100007-*xxx*; modem end is female. *xxx* represents cable length:

-001=5 feet
-002=10 feet
-003=25 feet
-004=50 feet
-005=70 feet

Step 2 Set the switches on the modem to select the correct communication parameters (refer to your modem manual for specific information). Table 4 through Table 10 list switch settings for the supported modems for standard dial-in or dial-out purposes, or for use with `tip` and UUCP.

Table 4 Incoming UUCP and dial-in settings, Trailblazer Plus and Trailblazer Plus/T2000

E1 F1 M1 Q0 P V1 X1 Version BA4.00

S00=001 S01=000 S02=043 S03=013 S04=010 S05=008 S06=002 S07=060 S08=002

S09=006 S10=007 S11=070 S12=050 S45=000 S47=004 S48=000 S49=000

S50=000 S51=004 S52=002 S53=003 S54=000 S55=000 S56=017 S57=019 S58=003

S59=000 S60=000 S61=045 S62=003 S63=001 S64=000 S65=000 S66=000 S67=000

S68=255 S90=000 S91=000 S92=000 S95=000

S100=000 S101=000 S102=000 S104=000 S110=255 S111=030 S112=001 S121=000

N0: N1: N2: N3: N4: N5: N6: N7: N8: N9:

Note: Enter S53=003 last, in the form of ATS53=003&W. Once this command is entered, you will not be able to use `tip` to directly access the modem without a breakout box.

Table 5 Incoming UUCP and dial-in settings for Racal-Vadic VA212

1*2	STANDARD OPTN	*1	ASYNCH/SYNCH
3*2	DATA RATE SEL	4*1	103 OPERATION
5*3	CHARACTER LEN	6*2	ORIG/ANS MODE
7*2	SLAVE CLOCK	8*2	DTR CONTROL
9*2	ATT/UNATT DISC	10*1	LOSS CXR DISC
11*2	REC SPACE DISC	12*2	SEND SPACE DIS
13*1	ABORT DISC	14*1	REMOT TST RESP
15*3	DSR CONTROL	16*1	CXR CONTROL
17*1	AUTO LINKING	18*3	ALB CONTROL
19*1	AUTO ANSWER	20*2	TERMINAL BELL
21*2	LOCAL COPY	22*2	DIAL MODE
23*2	BLIND DIAL	24*2	CALL PROGRESS
25*2	FAIL CALL SEL	26*9	AUTO REDIAL

Table 6 Incoming UUCP and dial-in settings for Maxwell Modem 1200VP

S0	ANSWER ON RING *	001
S1	RING COUNT	002
S2	ESCAPE CODE	043
S3	CARRIAGE RETURN	013
S4	LINE FEED	010
S5	BACK SPACE	008
S6	WAIT DIAL TONE	002
S7	WAIT DATA CXR	030
S8	PAUSE TIME COMMA	002
S9	CXR DETECT RESPONSE TIME	006
S10	LOST CXR HANG-UP DELAY	014
S11	NOT USED	
S12	ESCAPE CODE GUARD TIME	050
S13	NOT USED	
S14	BIT MAPPED OPTIONS *	
S15	NOT USED	
S16	TEST OPTIONS	
S17	NOT USED	
S18	TEST TIMER *	000
S19	NOT USED	
S20	NOT USED	
S21	BIT MAPPED *	
S22	BIT MAPPED *	
S23	BIT MAPPED *	
S24	NOT USED	
S25	N/A WITH 1200VP	
S26	N/A WITH 1200VP	
S27	N/A WITH 1200VP	

Inside the modem, you will need to set W4 to B, "DSR follows off-hook relay."*
After the switches are set, enter:

```
at&c1&w
```

(The `&c1` allows DCD to follow true carrier.) Once you enter this command, you will not be able to use `tip` to directly access the modem without a breakout box.

Table 7 Outgoing UUCP for Trailblazer Plus and Trailblazer Plus/T2000

E1 F1 M2 Q0 T V1 X1 Version BA4.00

S00=000	S01=000	S02=043	S03=013	S04=010	S05=008
S06=002	S07=060	S08=002			
S09=006	S10=007	S11=070	S12=050	S45=000	S47=004
S48=000	S49=000				
S50=000	S51=255	S52=002	S53=000	S54=001	S55=000
S56=017	S57=019	S58=003			
S59=000	S60=000	S61=045	S62=003	S63=001	S64=001
S65=000	S66=000	S67=000			
S68=255	S90=000	S91=000	S92=001	S95=002	
S100=000	S101=000	S102=000	S104=000	S110=001	S111=030
S112=001	S121=000				

N0: N1: N2: N3: N4: N5: N6: N7: N8: N9:

Table 8 Outgoing UUCP and `tip` settings for Racal-Vadic VA212

1*2	STANDARD OPTN	*1	ASYNCH/SYNCH
3*2	DATA RATE SEL	4*1	103 OPERATION
5*3	CHARACTER LEN	6*1	ORIG/ANS MODE
7*2	SLAVE CLOCK	8*2	DTR CONTROL
9*2	ATT/UNATT DISC	10*1	LOSS CXR DISC
11*2	REC SPACE DISC	12*2	SEND SPACE DIS
13*1	ABORT DISC	14*1	REMOT TST RESP
15*3	DSR CONTROL	16*1	CXR CONTROL
17*1	AUTO LINKING	18*3	ALB CONTROL
19*1	AUTO ANSWER	20*1	TERMINAL BELL
21*2	LOCAL COPY	22*2	DIAL MODE
23*2	BLIND DIAL	24*2	CALL PROGRESS

Table 9 Outgoing UUCP and `tip` settings for Maxwell Modem 1200VP

S0	ANSWER ON RING *	000
S1	RING COUNT	000
S2	ESCAPE CODE	043
S3	CARRIAGE RETURN	013
S4	LINE FEED	010
S5	BACK SPACE	008
S6	WAIT DIAL TONE	002
S7	WAIT DATA CXR	030
S8	PAUSE TIME COMMA	002
S9	CXR DETECT RESPONSE TIME	006
S10	LOST CXR HANG-UP DELAY	014
S11	NOT USED	
S12	ESCAPE CODE GUARD TIME	050
S13	NOT USED	
S14	BIT MAPPED OPTIONS *	
S15	NOT USED	
S16	TEST OPTIONS	
S17	NOT USED	
S18	TEST TIMER *	000
S19	NOT USED	
S20	NOT USED	
S21	BIT MAPPED *	
S22	BIT MAPPED *	
S23	BIT MAPPED *	
S24	NOT USED	
S25	N/A WITH 1200VP	
S26	N/A WITH 1200VP	
S27	N/A WITH 1200VP	

Inside the modem, you will need to jumper W4 to A*; this holds DSR high. After the switches are set, enter:

```
% at&c0&w
```

(The `&c0` forces DCD high.)

Table 10 `tip` settings for Trailblazer Plus and Trailblazer Plus/T2000

```
E1 F1 M1 Q0 T V1 X1 Version BA4.00
S00=000  S01=000  S02=043  S03=013  S04=010  S05=008
S06=002  S07=040  S08=002
S09=006  S10=007  S11=070  S12=050  S45=000  S47=004
S48=000  S49=000
S50=000  S51=255  S52=002  S53=000  S54=002  S55=000
S56=017  S57=019  S58=003
S59=000  S60=000  S61=045  S62=003  S63=001  S64=000
S65=000  S66=001  S67=000
S68=255  S90=000  S91=000  S92=000  S95=002
S100=000 S101=000 S102=000 S104=000 S110=255 S111=255
S112=001 S121=000
N0: N1: N2: N3: N4: N5: N6: N7: N8: N9:
```

Configuring a modem for dial-in

If you configure a modem for dialing in, you will not be able to use `tip` to access that modem without a breakout box. Configure breakout boxes as follows:

Step 1 Connect DCE to the modem.

Step 2 Make sure the following pins are in the off (open) position:

- Pin 6 (Data Set Ready)
- Pin 8 (Data Carrier Detected)
- Pin 20 (Data Terminal Ready)

All other pins should be in the on (closed) position.

Step 3 Jumper the following pins on the DTE side of the breakout box:

- Pin 4 (Request to Send)
- Pin 6 (Data Set Ready)
- Pin 8 (Data Carrier Detected)

Configuring software

Complete the following steps to configure ConvexOS so it can recognize the modem:

Step 1 Log in as the superuser.

Step 2 If you are using the modem for dial-out purposes, link the tty port to the alternate modem device file cua*n*, where *n* is one- or two-digit number. The device file name cua*n* is by convention only. The file name can be anything. For example, if your modem is connected to device tty07 and it is the first modem you installed, create a link by entering:

```
# ln /dev/tty07 /dev/cua0
```

Additional modems would use device files named cua1, cua2, and so forth.

Step 3 If you are using the device for UUCP purposes, change ownership of the alternate device file to `uucp` and change the group to `daemon` by entering:

```
# chown uucp /dev/cua0
# chgrp daemon /dev/cua0
```

Step 4 Check the /etc/gettytab file for an entry suitable for use by 1200-, 2400-, and 9600-baud modems. The /etc/gettytab file contains terminal line definitions (such as baud rate), and is read each time the `getty` process starts. An example of an entry for use with auto-select 1200-, 2400-, or 9600-baud modems is shown in Figure 25. When the break signal is received at connection time, the system cycles through these entries.

Figure 25 Example /etc/gettytab entry for 1200-, 2400-, and 9600-baud modems

```
#9600/2400/1200
B|B9600|9600-baud::nx=B2400:tc=9600-baud:
B2400|2400-baud::nx=B1200:tc=2400-baud:
B1200|1200-baud::nx=B9600:tc=1200-baud:
```

The /etc/gettytab file also contains entries that specify the terminal line to operate at single baud rates only (see Figure 26).

Figure 26 Example single-baud entry in /etc/gettytab

```
f|std.1200|1200-baud:\
        :fd#1:sp#1200:
```

Step 5 If /etc/gettytab does not contain a suitable entry for your modem, modify the file to create a custom entry. The format of this file is

name|*alternate_name*[|...]:
:*attribute*:*attribute*:

where

name	is a single-character entry name.
alternate_name	is an optional alternate name or list of names. Any of the names specified can be used in the *command* field of the /etc/ttys file to reference this entry. The names are separated by vertical bars (|).
attribute	specifies the attributes for a modem. Consult the gettytab(5) man page for more information when modifying this file.

Step 6 Add an entry in the /etc/termcap file for the modem. Figure 27 shows an example modem entry in this file.

Figure 27 Example modem entry in the /etc/termcap file

```
su|dumb|dialup|dialin::am:bl=^G:co#80:cr=^M:do=^J:nl=^J:
```

The format of this file is

name|*alternate_name*[|...]:
:*attribute*:*attribute*:

where

name	is a single-character entry name.
alternate_name	is an optional alternate name or list of names. Any of the names specified can be used in the *type* field of the /etc/ttys file to reference this entry. The names are separated by vertical bars (|).
attribute	specifies the attributes for the terminal type. Consult the termcap(5) man page before modifying this file.

Step 7 Set the characteristics of your communications port by editing fields in the /etc/ttys file:

- Set the command field to reflect the correct terminal line characteristics from the /etc/gettytab file (for example, B2400).
- Specify the type as `modem` to indicate that the device is a modem.
- If you are using the modem as a dial-out device, set the on/off field to `off` so `getty` is not executed and the login prompt is never displayed. If you are using the modem for dial-in or incoming UUCP purposes, set this field to `on`.
- Specify the port as `dialup` if it is a dial-in port. This requires users to enter a password when dialing in to this port.
- If you are using the modem for incoming UUCP, set the UUCP field. This allows only user IDs of uucp to log into the port.

An example /etc/ttys file with these changes is shown in Figure 28.

Figure 28 Example modem entries in the /etc/ttys file

```
cua0    "/etc/getty B2400"      modem    off
cua1    "/etc/getty B2400"      modem    on     dialup    uucp
```

Dial-out port

Dial-in port

In the example, tty01 is used as a dial-out port and tty02 is used as a dial-in port. The tty device files were renamed cua0 and cua1 respectively. For more information on the structure of this file, see the section in this chapter titled, "Adding terminals," and the ttys(5) man page.

Step 8 Reinitialize the file with the `on` command by entering:

```
% on -s
```

The `on` command effectively updates all tty ports.

Step 9 Using the `nu` program, add a user account with the following information to create a dial-in account:

User name:	dialin
User ID:	15 (default user ID for user dial-in)
Group:	guest (assign default group ID 31 to group guest)
Home Dir:	/tmp
Shell:	/bin/false

The dial-in account does not require a valid shell or a separate home directory. Refer to Chapter 7, "Setting up user accounts," on page 161, for details on how to set up user accounts.

Step 10 Assign a password for the dial-in account using the `nu` program. This is the extra password users must enter to access their accounts through the dial-in port (provided the port is marked as `dialup`). Refer to Chapter 7, "Setting up user accounts," on page 161, for details on how to assign account passwords.

Step 11 Add the word `dialin` to the /etc/ftpusers file. If you do not make this entry, a user can use `ftp` to gain nonprivileged access to a system on a network even if they do not have access to an account on that system. Figure 29 shows an example /etc/ftpusers file.

Figure 29 Example /etc/ftpusers file

```
dialin
nuser
uucp
```

Step 12 If you are using `tip`, modify the /etc/phones file to include the phone numbers of the systems to which you want to dial out. See Figure 30 for an example /etc/phones file.

Figure 30 Example /etc/phones file

```
tut          5555555
ra           4444444
cleo         3333333
hdqtrs         9,,101,222,3333,,,,555-1212
```

Step 13 If you are using `tip`, modify the /etc/remote files to include the phone numbers and attributes of the systems to which you want to dial out. See Figure 31 for an example /etc/remote file.

Figure 31 Example /etc/remote file

```
#General dialer definitions used below
#
dial1200|1200 Baud Vadic attributes:\
 :dv=/dev/cua0:br#1200:cu=/dev/cua0:at=hayes:du:
dial9600|9600 Baud Vadic attributes:\
 :dv=/dev/cua0:br#9600:cu=/dev/cua0:at=hayes:du:
dial300|300 Baud Vadic attributes:\
 :dv=/dev/cua0:br#300:cu=/dev/cua0:at=hayes:du:
#
# UNIX system definitions
#
UNIX-1200|1200 Baud dial-out to another UNIX system:\
 :el=^U^C^R^O^D^S^Q@:ie=#%$:oe=^D:tc=dial1200:
UNIX-9600|9600 Baud dial-out to another UNIX system:\
 :el=^U^C^R^O^D^S^Q@:ie=#%$:oe=^D:tc=dial9600:
cu300|UNIX-300b|300 Baud dial-out to another UNIX system:\
 :el=^U^C^R^O^D^S^Q@:ie=#%$:oe=^D:tc=dial300:
#
tip0|tip1200:tc=UNIX-1200:
cu0|cu1200:tc=UNIX-1200:
cu1|cu9600:tc=UNIX-9600:

hayes:dv=/dev/cua0,/dev/cua1:br#1200:
tbitout:dv=/dev/cua1:br#19200:
tbitin:dv=/dev/ttyd3:br#19200:
cua0:dv=/dev/cua0:br#1200:
cua1:dv=/dev/cua1:br#1200:
bigjim:dv=/dev/cua2:br#2400:
#
tut:pn=5555555:dv=/dev/cua0:br#1200:at=vadic:du:
ra:pn=4444444:dv=/dev/cua0:br#1200:at=vadic:du:
cleo:pn=@:dv=/dev/cua0:br#1200:at=vadic:du:
```

Punctuation in phone numbers signifies delays, such as waiting for a second dial tone. The punctuation characters used depend on the type of modem. The attributes of the /etc/remote file are:

`dv`	Device file to use for the tty port.
`el`	EOL marks (default is `NULL`).
`du`	Make a call flag (dial up).
`pn`	Phone numbers—An "@ =>" entry indicates to look for the phone numbers in the /etc/phones file; possibly taken from PHONES environment variable).
`at`	Automatic Call Unit (ACU) type.
`ie`	Input EOF marks (default is `NULL`).
`oe`	Output EOF string (default is `NULL`).
`cu`	Call unit (default is `dv`).
`br`	Baud rate (defaults to `300`).
`fs`	Frame size (default is `BUFSIZ`)—Used in buffering writes on receive operations
`tc`	To continue a capability (must be last entry).

Step 14 If you are using `tip`, add the /usr/lib/uucp/LCK directory if it does not already exist. See Figure 32 for the commands to add this directory.

Figure 32 Adding the /usr/lib/uucp/LCK directory

```
# cd /usr/lib/uucp
# UUCP_SETUP
```

Adding a printer

The procedure for adding a printer depends on the type of controller on which the printer is attached. Complete the appropriate section for the type of controller you are using to install your printer.

Adding a serial printer

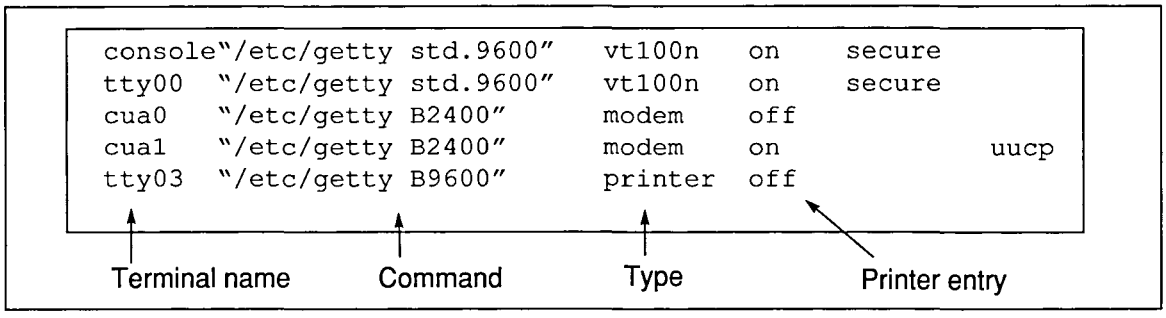
If you are adding a serial printer to an existing tty line, complete the following steps:

Step 1 Install the physical hardware.

Step 2 Log in as the superuser.

Step 3 Edit the entry in the /etc/ttys file for the terminal line to which you attached the printer. Set the `type` field to `printer` and the `on/off` field to `off`, as shown in Figure 33.

Figure 33 Example serial printer entry in /etc/ttys file



Step 4 Edit the /etc/printcap file to include a description of the printer. Refer to Chapter 5, "Setting up the line printer system," on page 129, for a description of the /etc/printcap file and how to edit it.

Adding a printer to PRC controller

If you are adding a printer to an existing PRC printer controller or are adding a new PRC controller, complete the following steps:

Step 1 Install the physical hardware. If this is an unsupported device, you must install a device driver. Refer to the *CONVEX Guide to Writing Device Drivers* for information on writing and installing drivers.

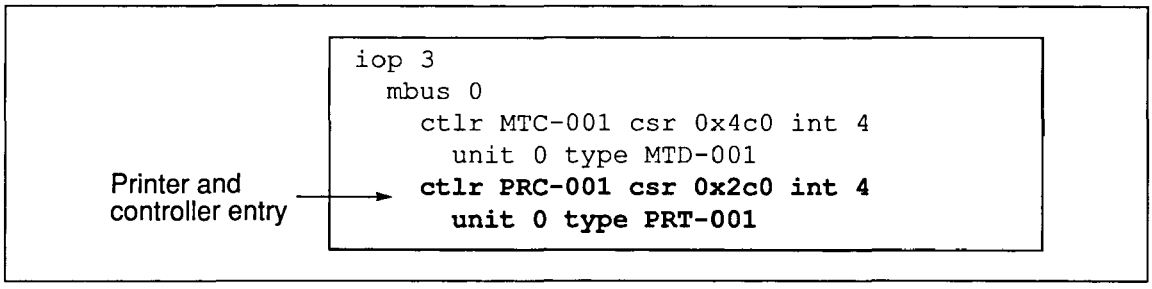
Step 2 Power up the system and boot to the SPU level. Information on powering up and booting the system can be found in the *CONVEX SPU System Manager's Guide* or the *CONVEX C3800 Series SPU System Manager's Guide*, whichever is available at your site.

Step 3 Make a back-up copy of the /ioconfig file on the SPU. Enter:

```
(spu)> cp /ioconfig /ioconfig.old
```

Step 4 Edit the /ioconfig file to include information about your device. The amount of information you add depends on the hardware you installed. Figure 34 illustrates adding a new printer controller and printer to an existing Multibus.

Figure 34 Adding a new printer controller and printer on existing Multibus



Refer to the section, "The /ioconfig file," on page 22, for information on the structure of this file.

Step 5 Change to the /mnt/os directory. Enter:

```
(spu)> cd /mnt/os
```

Step 6 Boot the system by entering:

```
(spu)> boot
```

Step 7 Log in as the superuser.

Step 8 Edit the /etc/printcap file to include a description of the printer. The /etc/printcap file is a database that describes various characteristics of a printer. Refer to Chapter 5, "Setting up the line printer system," on page 129, for a description of the /etc/printcap file and how to edit it.

Step 9 Check the /etc/printcap file for an lp device entry. Each printer installed in your system must have a device entry. Line printer device entries begin with /dev/lp0, and are numbered in increments of 8. lp0 has a default entry, but you must add entries for other lp devices.

Step 10 If necessary, create a special device file for the device using the MAKEDEV script. For example, create a device entry for a second printer by entering:

```
# cd /dev
# MAKEDEV pa1
```

MAKEDEV creates the file lp8 in the /dev directory.

Step 11 Change the the file created in Step 10 so that the owner of the file is lpr, the group ownership is lpr, and the access mode is 660.

The following example changes these ownerships and modes for the /dev/pa1 file:

```
# chown lpr /dev/pa1
# chgrp lpr /dev/pa1
# chmod 660 /dev/pa1
```

Adding a plotter

Complete the following steps to install a plotter:

Step 1 Install the physical hardware. If this is an unsupported device, you must install a device driver. Refer to the *CONVEX Guide to Writing Device Drivers* for information on writing and installing drivers.

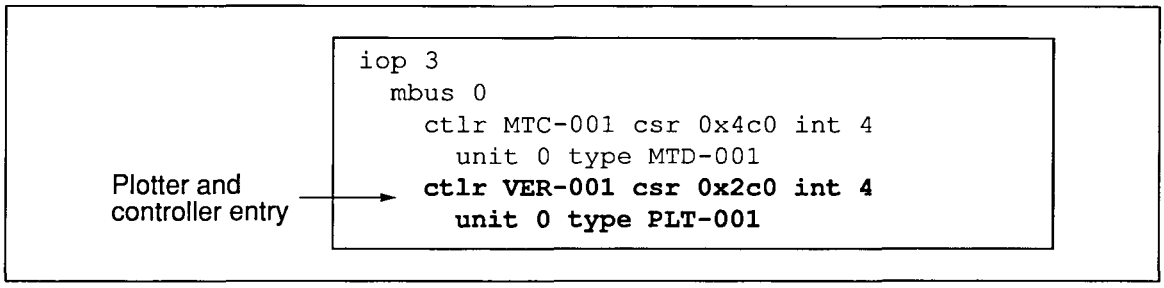
Step 2 Power up the system and boot to the SPU level. Information on powering up and booting the system can be found in *CONVEX SPU System Manager's Guide* or the *CONVEX C3800 Series SPU System Manager's Guide*, whichever is available at your site.

Step 3 Make a back-up copy of the /ioconfig file on the SPU. Enter:

```
(spu)> cp /ioconfig /ioconfig.old
```

Step 4 Edit the /ioconfig file to include information about your device. The amount of information you add depends on the hardware you installed. Figure 35 illustrates adding a new description for a plotter controller and Versatec plotter to a Multibus.

Figure 35 Adding a plotter controller and Versatec plotter to a Multibus



Refer to the section, "The /ioconfig file," on page 22, for information on the structure of this file.

Step 5 Change to the /mnt/os directory. Enter:

```
(spu)> cd /mnt/os
```

Step 6 Boot the system by entering:

```
(spu)> boot
```

Step 7 Log in as the superuser.

Step 8 Edit the /etc/printcap file to include a description of the plotter. The /etc/printcap file is a database that describes various characteristics of printers and plotters. Refer Chapter 5, "Setting up the line printer system," on page 129, for a description of the /etc/printcap file and how to edit it.

Step 9 If necessary, create a special device file for the device using the MAKEDEV script. For example, create a device entry for plotter 0 (zero) by entering:

```
# cd /dev
# MAKEDEV pb0
```

MAKEDEV creates the file pb0 in the /dev directory.

Setting up the disk system 3

Setting up your disk system is a multilayered task. Before you can configure disk partitions, you must plan the use of your disks. However, before you can plan effective use of your disks, you must understand certain concepts associated with the disks and file systems.

This chapter presents information for setting up a disk system. The major topics in this chapter are presented in the following order:

- Understanding disk and file system concepts
- Planning your disk system
- Configuring partitions and swap space

This order and presentation allows you to use the information according to your previous experience with setting up disk systems in the ConvexOS environment. For example, if this is the first time you are setting up your disk system, read all three topics in the order they are presented. If you already understand disk system concepts, skip the section on concepts and start with the section on planning. If you have already planned your disk system and are merely making changes to the partitions, skip to the section on configuring partitions.

Understanding disk system concepts

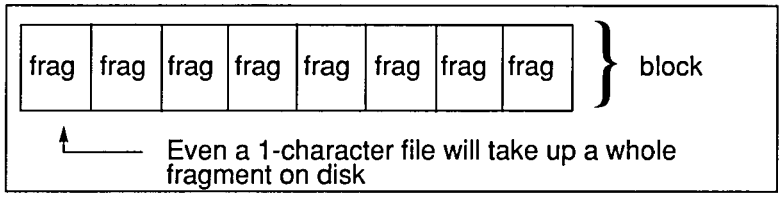
To set up your disk system, you must be familiar with some basic ConvexOS concepts, such as the units used to map disk space, different ways of arranging disk partitions, disk naming conventions, and load balancing. Each of these concepts is described in detail in the following sections of this chapter.

Mapping disk space

Blocks and fragments are concepts basic to disk storage under ConvexOS. You use these elements in tuning your disk system to fit your site's needs.

A block is the maximum amount of contiguous data that can be transferred to or from a disk as a unit. Blocks are made up of fragments. A fragment is the minimum amount of disk space used by a file. Figure 36 illustrates this relationship.

Figure 36 Block and fragments



Block size can be 4, 8, 16, 32, or 64 kbytes. Fragment size is a ratio of block size. It can be 1/8, 1/4, 1/2 of, or equal to block size. (The minimum fragment size for an IDC disk is 2 kbytes.) The maximum ratio of block size to fragment size is 8:1. Table 11 lists possible fragment sizes for the available block sizes.

Table 11 Block and fragment sizes

If block size is	Fragment size can be (kbytes)
4 kbytes	1/2, 1, 2, or 4
8 kbytes	1, 2, 4, or 8
16 kbytes	2, 4, 8, or 16
32 kbytes	4, 8, 16, or 32
64 kbytes	8, 16, 32, or 64

Disks can be divided into slices known as partitions. Each partition can have a different block and fragment size.

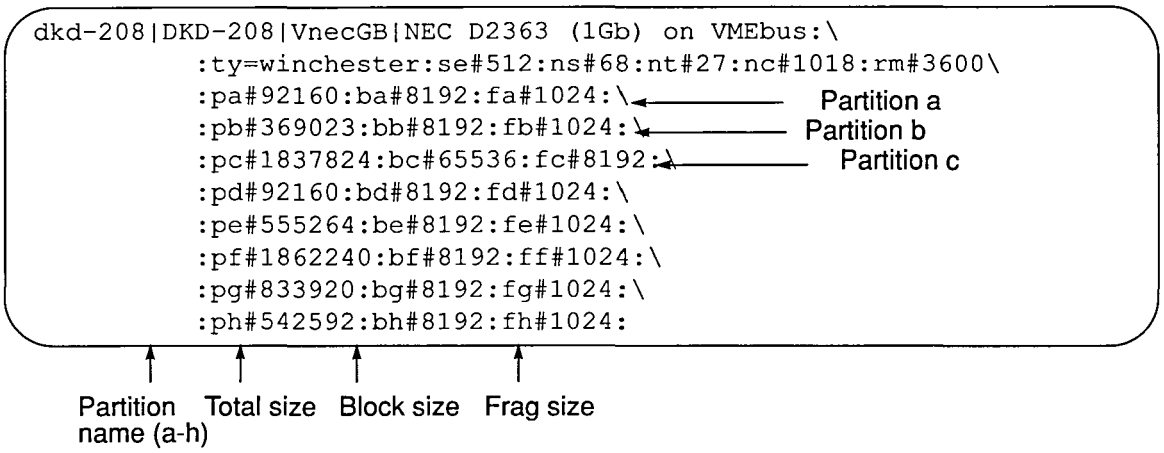
Disk partitions

A disk partition is a contiguous area of a disk device configured as a logical unit. Each partition can contain one regular file system or a portion of a striped file system (more on file systems in the next section).

Under ConvexOS, disks can be configured as one partition or can be divided into four or six partitions. A disk configured with more than one partition logically appears as multiple disks. This allows you to locate more than one file system on a disk, because each partition can contain one file system.

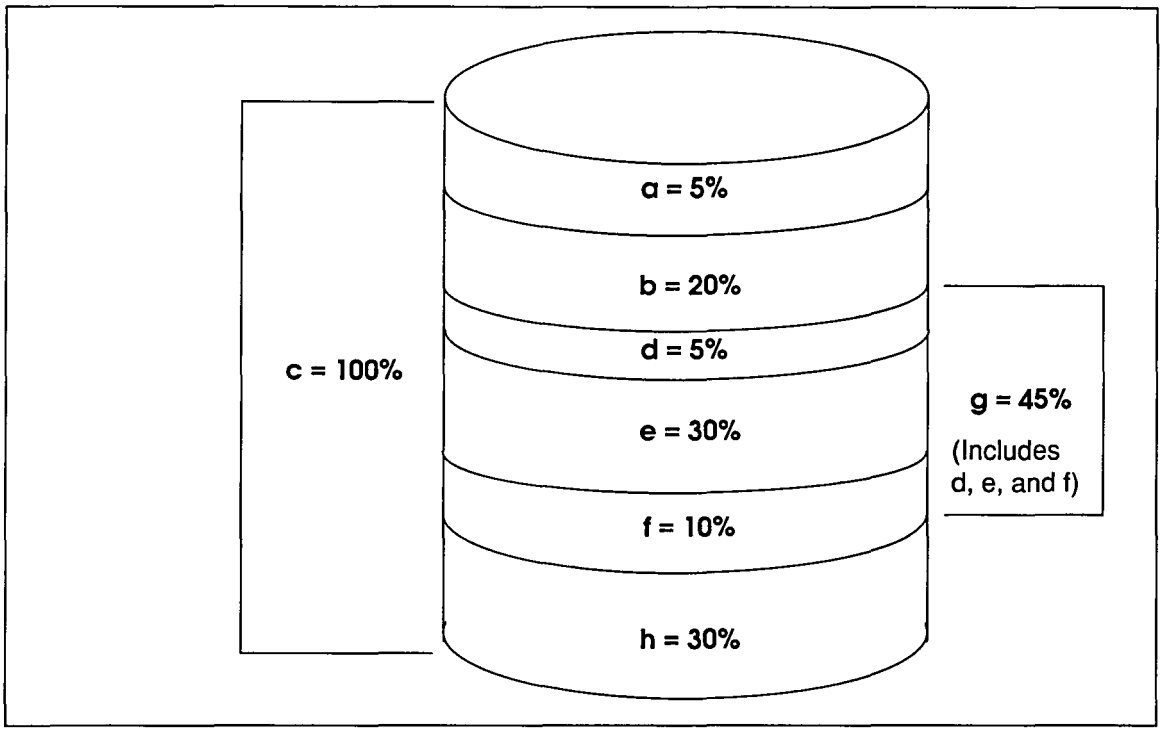
ConvexOS has predefined sizes and names for disk partitions. These are shown in the example /etc/disktab file in Figure 37. You cannot change the size or location of these partitions; you can change the block and fragment sizes for a partition (also shown).

Figure 37 Disktab file



The eight available partitions have preassigned names of *a*, *b*, *c*, *d*, *e*, *f*, *g*, and *h*. Each partition is a different area of the disk and has a preassigned percentage of the disk it can use. For example, partition *e* allocates 30% of a disk, and partition *g* allocates 45% of a disk. Figure 38 illustrates the area and percentage of disk used by each partition.

Figure 38 Preassigned partition percentage allocations



Caution

Do not overlap partitions that are preassigned to the same area.

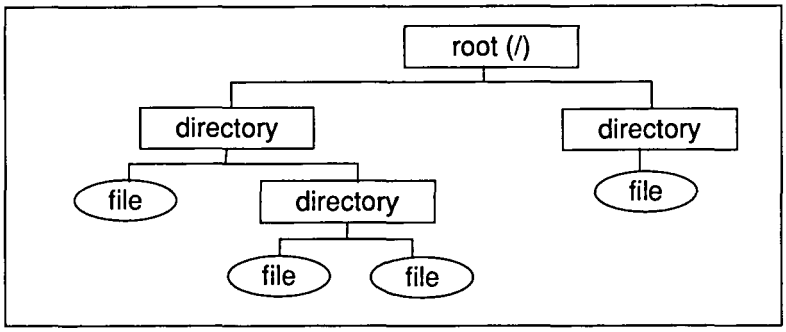
For example, in Figure 38, partition *g* is assigned the same area as partitions *d*, *e*, and *f* collectively. This means that if you assign partition *g* of a disk, you must not assign partitions *d*, *e*, or *f* of the same disk. And, if you assign partition *c* of a disk, you must not assign partitions *a*, *b*, *d*, *e*, *f*, *g*, or *h* of that disk.

Partitions *a* and *b* of the first disk (disk 0) have a default use. The *a* partition of the first disk is always the default / (root) file system, which includes the /etc, /bin, and /dev directories. The *b* partition of the first disk is the default swap partition. Although you can configure additional partitions on other disks to be swap partitions also, CONVEX recommends that partition *b* never be allocated to anything other than swap space.

ConvexOS file systems

File systems are structures that computers use to organize and store information. In ConvexOS, the file system is a hierarchical tree as illustrated in Figure 39.

Figure 39 File system hierarchical tree



Each file is connected or related to another starting with the root (/) directory and continuing downward through a virtually unlimited number of files. Directories provide an entrance to the next level in the hierarchical file structure. They can contain other directories, ordinary files, or nothing at all. Ordinary files contain data. They can have no other files beneath them and so are always the last file in a path.

The term file system can refer to the entire file tree or a subsection of the file tree. If it refers to a subsection of the tree, it references a collection of files, directories, and file management structures, such as inodes (index nodes) and superblocks located on a mass storage device.

Because many system maintenance tasks, such as allocation of disk space and dumping to tape, are done on a per-file-system basis, files are grouped in file systems to make performing these tasks easier. For example, by placing static system files and the more volatile user files in separate file systems, you can dump user files on a regular basis without also dumping system files that have not changed.

ConvexOS is delivered with the following four file systems:

/ Contains the / (root) file system and a minimal set of directories and ordinary files needed by the operating system to function correctly.

/tmp Contains temporary files created by editors and system processes.

/usr Contains user and system administrator commands and supporting information needed by users, such as libraries and spooling directories.

/mnt Contains users' home directories and their files.

The standard subdirectories of root and their uses are listed below:

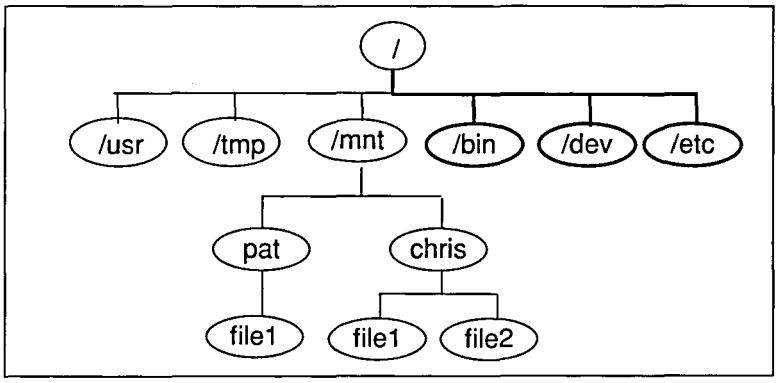
/bin Contains ConvexOS utility programs needed in single-user mode.

/dev Contains all files for peripheral devices, such as tape drives, disk drives, printers, and modems.

/etc Contains critical system files and maintenance programs.

Figure 40 illustrates the standard file system hierarchical structure. Directories on the root file system are shown in boldface.

Figure 40 Standard file system hierarchical structure



File systems defined in the directory structure correspond to partitions, regular or striped, on the physical disk. You determine which file systems are best suited to what sizes and types of disk partitions when you plan your disk system.

Disk space can be assigned to a file system in one of four different partition configurations. The first and most common is the single disk partition, which is discussed in the "Disk partitions" section of this chapter. The remaining configuration types are swap space, striped file systems, and redundant striped file systems.

Swap space

ConvexOS uses swap space to move data from main memory to disk until the process using the data becomes active again. Data transfers from memory to swap space and back happen faster if multiple disks are used for swap space.

Swap space functions like other file systems, but is used by ConvexOS rather than by users. By default, swap space is located on partition *b* of disk 0 (d*x*0b). However, you can designate any disk partition as additional swap space.

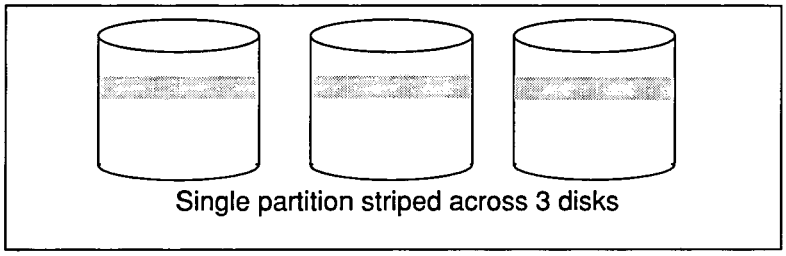
Striped partitions

A file system is restricted to the size of its partition. Thus, with conventional partitioning, a file system cannot be larger than the disk storage device on which it is located.

Disk striping allows you to combine multiple physical disk partitions into one logical partition; the combined partitions are logically treated as a single partition. This allows a file system to span multiple disk drives. However, the total size of a striped partition cannot exceed 1 terabyte minus 512 bytes.

Disk striping is most effective in a multiple disk system such as that in Figure 41, where the striped partitions are not on the same disk or on the same controller. Although you are not restricted to this scenario, striping across multiple disks increases throughput performance.

Figure 41 Disk striping



The system default is a maximum of 16 stripes. You can increase the maximum to as much as 256 by setting the `stripe_devices` boot-time parameter. For details on how to do this, refer to Chapter 15, "Customizing kernel boot-time parameters," on page 287.

In a multiple disk system, striping provides several benefits:

- Allows creation of a file system larger than one drive. You can combine two or more partitions (up to 128) across multiple disks to create a single striped file system.
- Increases performance by balancing the disk load. Data in striped file systems is read and written concurrently on all disk drives in the striped file system, so I/O processing is spread across all available resources.

Redundant striped partitions

One drawback of normal disk striping is that loss of any disk in a stripe causes the loss of all data in that stripe. Redundant striping schemes solve this problem, providing a means to restore or reconstruct your data in the event of disk failure.

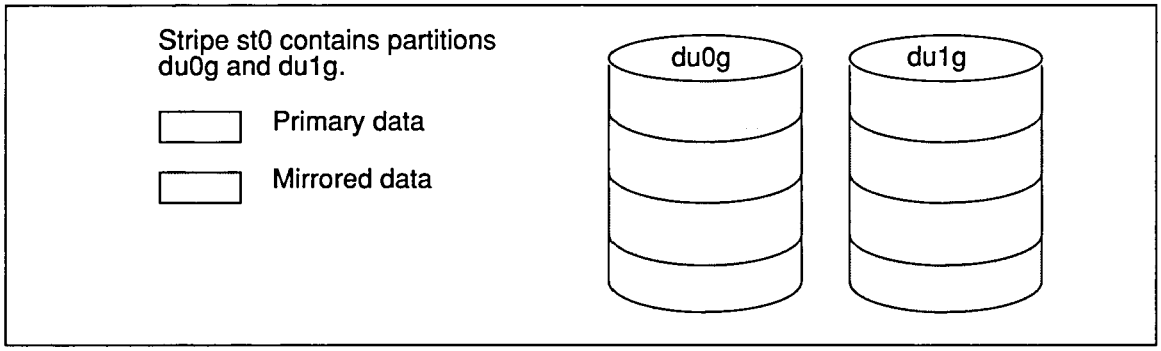
Two types of redundant file systems are available: mirrored and parity.

- Mirrored file systems maintain a copy of each stripe partition on a different disk. If the primary disk fails, a copy of its data can be retrieved from the second disk.

  Mirroring is the most reliable form of data redundancy. However, because redundant data occupies half of a mirrored stripe, mirroring is costly in terms of disk space.

  Redundant stripes containing only two disks are mirrored by default. To force mirroring for other stripes , specify `-P2` on the `newst` command line when you create the stripe. (The "Configure disk partitions" section of this chapter explains use of the `newst` command.) Figure 42 shows a mirrored stripe containing two disk partitions. Each band represents one stripe section.

Figure 42 Redundant stripe using mirroring



- Parity file systems use one partition in each stripe section to store parity information calculated from the data on the other disks in that stripe. If one of the disks in the stripe fails, the data on that disk can be reconstructed using the parity information.

  Parity is less reliable than mirroring, although data is lost only if two or more disks in a stripe fail at the same time. The amount of storage space required for parity information depends on the layout of your stripe. To determine the space used by parity information, issue the `newst` command to

create the desired stripe, including the -n option as shown in Figure 43. This command generates stripe information without actually creating the stripe.

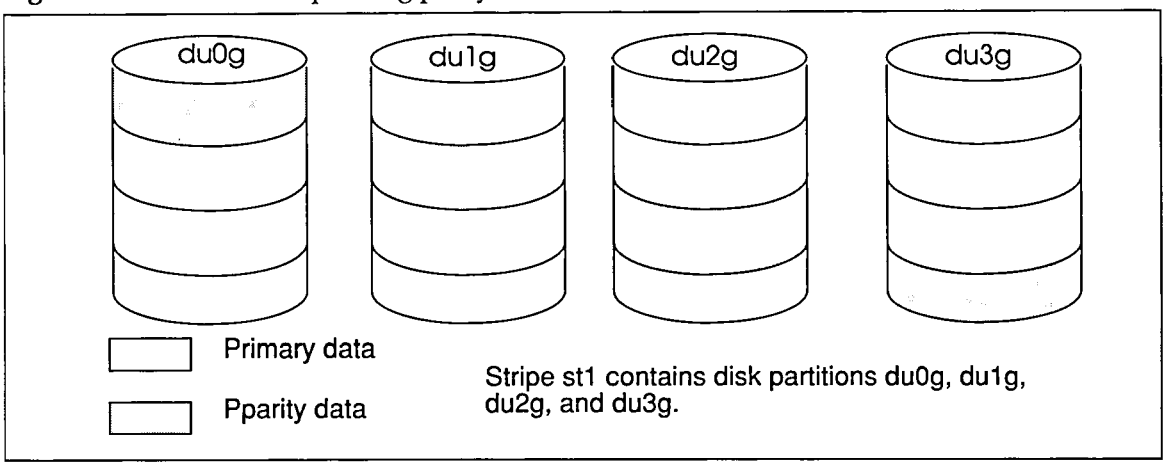
Figure 43 Full output from newst command

```
# /etc/newst -nvR st0 du0c dkd-502 du1c dkd-502 du2c dkd-502 du3c dkd-502 \
du4c dkd-502 du5c dkd-502
stripe st0: redundant, sector size 2048 bytes
        section a: size 489296 Kbytes/partition, blocking factor 16 Kbytes
                partition 0: du5c (64, 1283) offset 0 Kbytes
                partition 1: du1c (64, 259) offset 0 Kbytes
                partition 2: du2c (64, 515) offset 0 Kbytes
                partition 3: du3c (64, 771) offset 0 Kbytes
                partition 4: du4c (64, 1027) offset 0 Kbytes
        section b: size 489296 Kbytes/partition, blocking factor 16 Kbytes
                partition 0: du0c (64, 3) offset 0 Kbytes
                partition 1: du1c (64, 259) offset 489296 Kbytes
                partition 2: du2c (64, 515) offset 489296 Kbytes
                partition 3: du3c (64, 771) offset 489296 Kbytes
                partition 4: du4c (64, 1027) offset 489296 Kbytes
        section c: size 489296 Kbytes/partition, blocking factor 16 Kbytes
                partition 0: du0c (64, 3) offset 489296 Kbytes
                partition 1: du5c (64, 1283) offset 489296 Kbytes
/etc/putst /dev/rst0
newst: warning, 'size' & 'cpg' mkfs args are estimates, due to the '-n' option
/etc/mkfs /dev/rst0 2201832 180 7 65536 8192 4 1 32 10 60 2048 60
/etc/fsirand /dev/rst0
# ▮
```

Add the kbytes/partition shown in the command output for each section in the stripe. For sections a, b, and c in the example above, the space used by parity information is 489296 + 489296 + 489296 = 1467888 kbytes. The space available for data in each section is *xxxx*(*n*-1) kbytes, where *xxxx* is the kbytes/partition for that section, and *n* is the number of partitions in the section.

Any redundant stripe containing more than two disk partitions automatically uses parity unless forced to mirror by the newst -P2 option. Figure 44 shows a redundant stripe using parity. Each band represents one stripe section.

Figure 44 Redundant stripe using parity

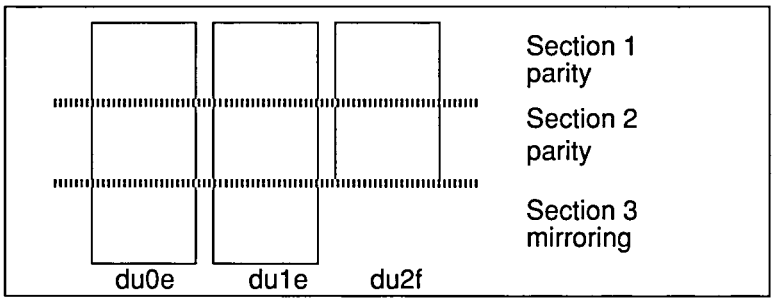


Stripe sections

As illustrated in Figure 42 and Figure 44, redundant stripes are divided into sections. The `newst` utility determines the distribution of the sections across disk partitions in the stripe.

`newst` arranges stripe sections to make the most efficient use possible of the available space. In some cases, this leads to stripes that use mirroring for some sections and parity for others. For instance, a stripe containing partitions du0e, du1e and du2f might be sectioned as shown in Figure 45.

Figure 45 Stripe sections



For a breakdown of the sections in any stripe, use the `getst` command, with the following option:

`getst st`*xx*

where `st`*xx* is the stripe in question.

Hot spares

When you set up your disk system, you can designate certain disk partitions as hot spares. Disks designated as hot spares can be moved into redundant stripes as replacements for partitions of failed disks.

If hot spare disk space is available when a disk in a redundant stripe fails, `vvmdaemon` attempts to reconstruct the data from the failed disk onto a designated hot spare. The hot spare replaces the failed disk in the stripe, and operations on the stripe continue as before.

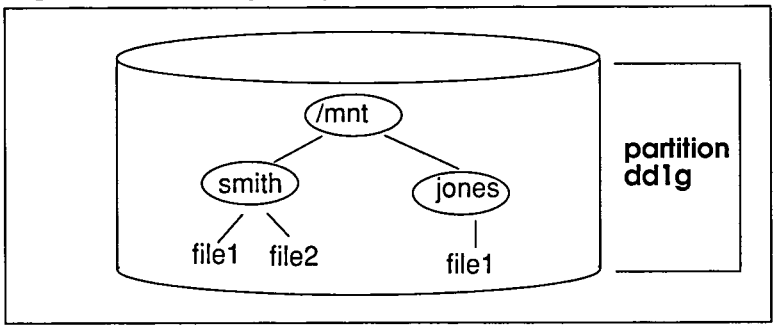
If a hot spare is not available, reads and writes (nonredundant) to the stripe continue. Performance degrades somewhat until you replace the failed disk.

If hot spare disk space is not available when a disk fails, console error messages direct you to reconstruct the data manually. Appendix C of *Managing ConvexOS: Operations Guide* explains the manual reconstruction procedure.

Mount points

Each partition of a disk may contain one nonstriped file system or a portion of a striped file system. Each file system contains a set of directories and files that are physically located on the disk within the partition or stripe boundaries. For example, in Figure 46, the /mnt directory and all its associated files physically reside on partition *g* of disk drive dd1. This partition has a block device name of dd1g.

Figure 46 Partition *g* file system



File systems (located on partitions) are attached to the file tree by mounting them using the `mount` command. The directories and files in a file system are not accessible until the file system is mounted.

Note

The root file system is mounted by ConvexOS at boot time and cannot be unmounted.

File systems are mounted to a directory in the file tree. The directory where you want to mount the file system must already exist. The directory on which a file system is mounted is called a mount point. For example, Figure 47 illustrates the file hierarchy of a computer system before mounting the file system named /dev/dd1g.

Figure 47 Example file tree before mounting dd1g

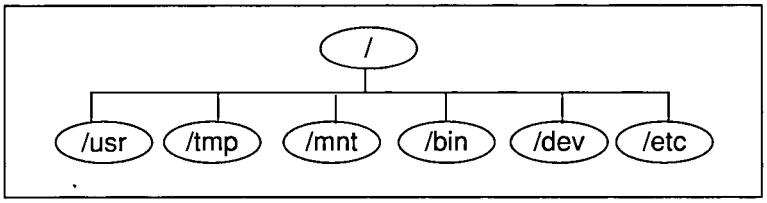
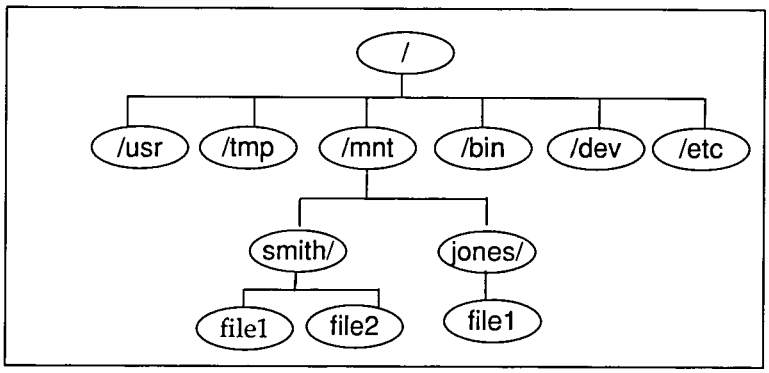


Figure 48 illustrates the file hierarchy after mounting /dev/dd1g on the /mnt directory.

Figure 48 Example file tree after mounting dd1g



In this case, the /mnt directory is the mount point of the file system referenced as /dev/dd1g. The `mount` command tells the operating system that the /mnt mount point is now equivalent to the top level directory of the file system.

Note

Make sure directories that serve as mount points have mode 777 so access problems do not occur.

Mount-point directories are usually empty. If they do contain files, those files become inaccessible when a file system is mounted because the contents of the mounted file system hide the true contents of the directory. The name of the mount-point directory does not change.

The contents of the mount point are unavailable for as long as a file system is mounted there, but are not affected by the mounting process. Once the /dev/dd1g file system is unmounted using the `umount` command, any files in the /mnt directory can be accessed again.

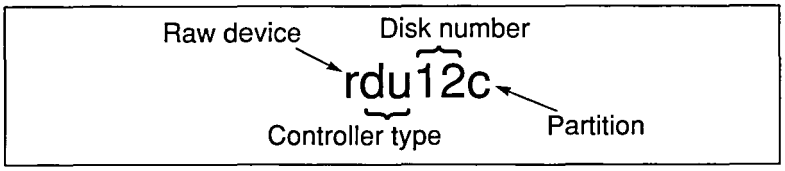
A file system may be mounted on any directory in the hierarchy tree that is not already busy. A directory is busy when it is the current directory or parent of the current directory of a process or has been opened for reading (such as the `ls` command) or writing.

Although not the case in the example above, /mnt and /mnt/smith can be completely separate file systems, even though /mnt/smith is mounted under /mnt.

Disk, partition, and stripe naming conventions

When setting up your disk system, you must be able to interpret and assign logical names to the disks, partitions, and stripes in your system. For disk and partitions, the name indicates whether the named device is a raw device or a block device, what type of controller it is connected to, the number assigned to the disk, and the partition name (if you are naming a partition), as shown in Figure 49.

Figure 49 Disk device naming scheme



Raw device names begin with an *r*, and support direct I/O to the disk drive. Block device names omit the *r*, and support I/O to the disk drive through a buffer cache. All disk drives have both raw and block device device names.

The next two characters of the partition or disk name specify the type of controller to which the disk is attached. This can be one of the following:

da = Multibus controller

dd = VMEbus controller

du = IDC controller

The next one or two characters of the partition or disk name specify the unique number assigned to the disk. This number is based on the disk's occurrence in the ioconfig file located on the SPU. See the section in this chapter titled "Planning your disk system" for more details on numbering disks.

If the named device is a partition, the next character specifies the disk partition. This can be any letter between *a* and *h*.

Stripes have similar naming conventions. For example, rst3 is the raw device for stripe 3. The *r* indicates a raw device, *st* identifies it as the name of a stripe, and the 3 indicates its stripe number.

Stripes are numbered consecutively according to their chronological inclusion in the system, beginning with the number 0. Stripes do not have a disk partition letter associated with their name, as do regular disk devices.

Disk load balancing

Disk load balancing means distributing file system use across available disks and disk controllers. A balanced load is critical in multiple disk configurations when the system is busy. If all disk activity at one time needs the resources of the same disk, throughput is limited to the bandwidth of the disk. When you evenly distribute the most often used file systems across disks, disk performance improves, often doubling throughput.

CONVEX provides two ways to improve disk load-balancing: adding disks and disk striping. The following sections describe the benefits and considerations for these methods.

Adding disks

Adding disks and controllers increases flexibility and makes loadbalancing easier. Ideally, systems that contain several buses and several channel control units (CCUs) are configured to have each physical disk on a separate bus, a separate controller, and preferably separate CCUs.

Striping disks

Disk striping allows file systems to be distributed across multiple disks. Although disk striping is usually desirable, it increases the risk of data loss during a hardware failure. In a redundant striped file system spanning several disk drives, failure of one drive renders the striped file system unusable until the hardware failure is corrected and the file system is re-created from an archived backup.

Redundant striping protects against such data loss and system downtime. However, redundant stripes require extra disk space.

Even though disk striping potentially increases disk performance, bad disk-striping decisions make disk performance worse. Consider these performance guidelines when deciding on disk striping:

- Partitions should span two or more disks.

  If you stripe two or more partitions on the same disk, the disk arm alternates from one partition to another on the same disk between each read. A long seek operation is required on each successive read, seriously impairing performance.

- Partitions should span multiple controllers.

  When partitions of a disk stripe span multiple controllers, write requests issued by each controller to each disk occur

simultaneously. This is called overlapping and provides the best performance.

If two or more partitions of a disk stripe are on the same disk controller, operations are performed in sequence instead of overlapped, negating the benefits of disk striping. For example, if you stripe three disk partitions on one disk controller and a fourth on a second disk controller, the best throughput is only one and 1/2 times better than on a file system that is not striped.

- Avoid striping across different types of disk drives because you lose the performance benefits of the faster bus. For example, avoid striping a VMEbus disk with an IDC disk.
- Partitions should span buses.

  There is insufficient bandwidth on a single bus to allow more than two disk controllers to make simultaneous data transfers. If one stripe is on three or more controllers using the same bus, one or more controllers must delay transfer of data, limiting disk performance.
- In configurations with Multibus disks, VMEbus disks, and IDC disks, put the most active file systems on VMEbus and IDC disks because they are faster drives.
- If possible, stripe together partitions of the same size and type for better performance.
- Never stripe the / (root) file system. The root file system should be on the *a* partition of the first disk (disk 0).
- Because swap space is allocated in the kernel, never include swap partitions in a disk stripe. All swap space is grouped for enhanced read/write performance.
- The stripe width of a redundant stripe using parity should be some power of two $(2^n) + 1$. Because block sizes are powers of two, a stripe width of $2^n + 1$ allows for the most efficient read/write of one block of data (2^n) plus parity information (+ 1). Stripe widths that do not follow this formula cause a loss of read/write performance because a block of data cannot be evenly distributed.

Planning your disk system

Setting up your disk system means allocating file systems to available disks. However, it is not as simple as that. When allocating disk space, you must consider many aspects, including available resources and system performance.

The physical resources of your system, such as the amount of memory and the number of available disks and controllers, affect performance of the disk system. Adding resources increases performance, as outlined below:

- The larger the buffer cache, the better your disk performance, especially when processing large data files. The buffer cache is an area of physical memory allocated to buffering data from disk.

  Memory used for the buffer cache comes from the general pool of memory. The amount of memory used as buffer cache versus that used for programs is dynamically balanced by the kernel.

- Adding physical memory enlarges the buffer cache without taking away memory for user programs. The maximum amount of memory allocated for the buffer cache is the smaller of:
  - 90% of the amount of memory available at boot time
  - 8192 times the size of a file system buffer, with a maximum limit of 512 megabytes

- The number of disks and disk controllers affects performance as well. More disks and controllers allow more flexibility in distributing disk-system load. That is, disk-system resources are used more efficiently because the processing load is balanced.

  System performance also depends on some system administrator-defined variable parameters in the disk system that affect speed and space consumption, such as block and fragment sizes and disk striping. Tune block and fragment sizes for the needs of your installation when you create a file system.

  Disk striping allows you to combine multiple disk partitions into one logical device so the combined partitions are treated as a single partition. See the "Striped partitions" section in this chapter for more details on disk striping.

Achieving good system performance involves compromises of all of these aspects. Changing one variable affects others. For example:

- Establishing a larger block and fragment size allows faster transfer of data but can decrease space efficiency because even small files take up a full fragment on disk.
- Allocating more resources to one file system adversely affects other file systems.
- Creating large disk stripes increases space and speed. However, redundant stripes increase the risk of data loss because if one disk fails, you lose the entire stripe partition. Redundant stripes protect against data loss, but use extra disk space.

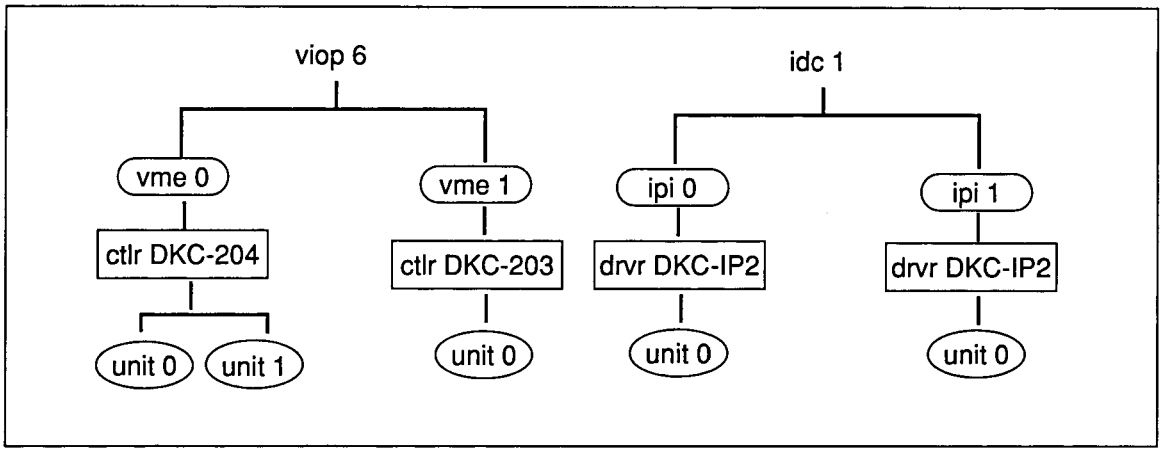
Depending on your system configuration and use, the issues of speed, space efficiency, load balance, and data loss can range from unimportant to extremely important. For example, if you have as many disks and controllers as you need, space efficiency is much less of a concern than speed.

If you regularly back up your file systems to tape, the data-loss risk associated with nonredundant disk striping may not be very important. With a one-disk system, load balancing is not applicable. With larger systems, load balancing increases performance.

The following paragraphs discuss in detail disk system planning issues and present procedures to resolve them.

Step 1 Make a diagram similar to the one shown in Figure 50 of the disks on your system; include the bus and controller connected to each disk.

Figure 50 Disk configuration diagram

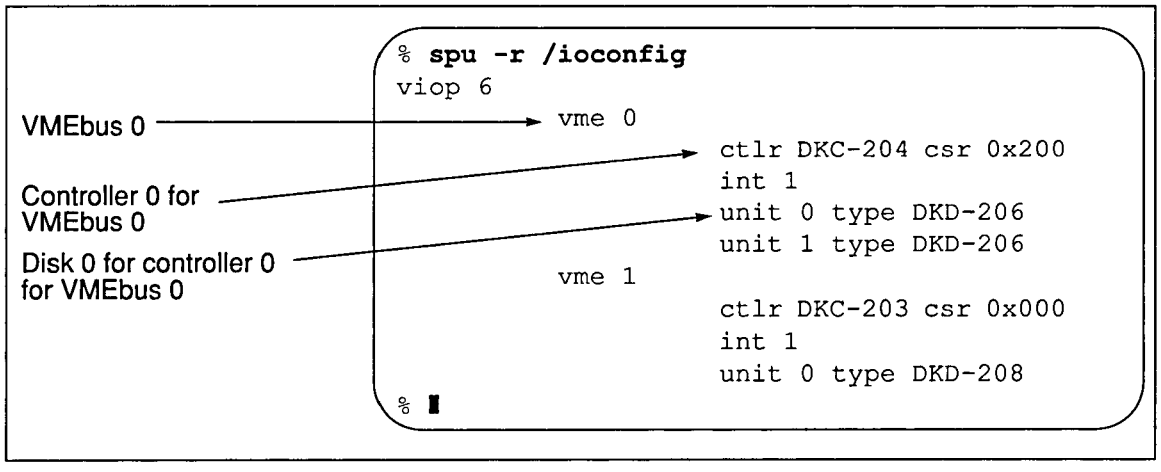


This information is available in the ioconfig file located on the SPU. To display the contents of the ioconfig file, enter

```
# spu -r /ioconfig
```

The ioconfig file shows each communication channel, controller, and disk included in your system. Figure 51 illustrates example output for this command.

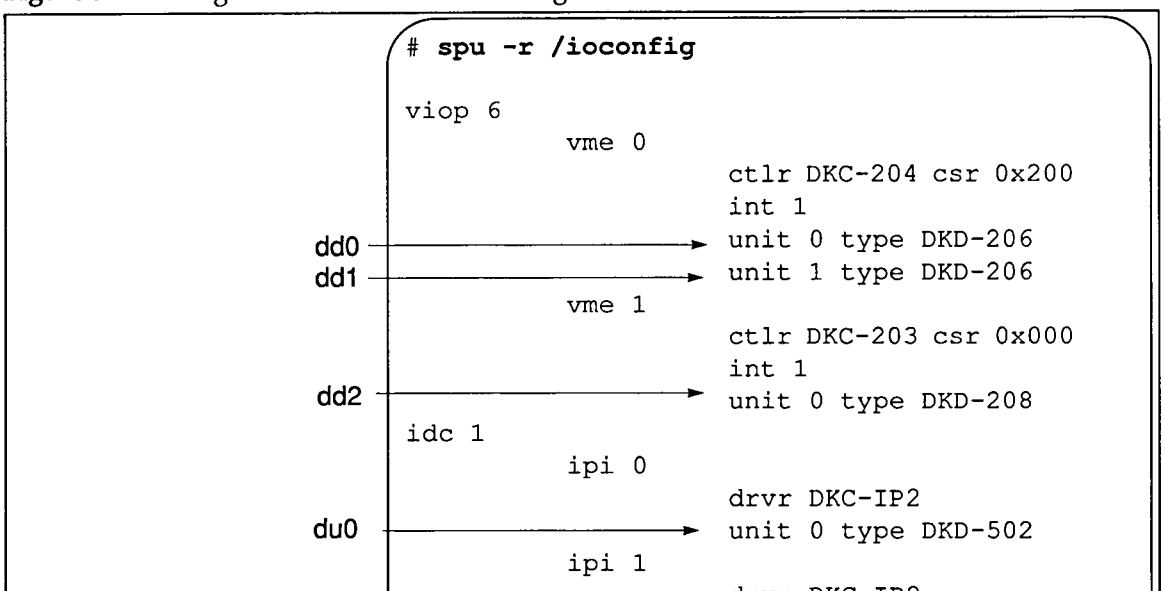
Figure 51 Sample ioconfig file



Step 2 Assign numbers to your disks.

The ioconfig file on the SPU lists the physical devices for your system. In the ioconfig file, disks (units) are numbered sequentially starting with 0 for each new controller. For example, in the ioconfig file shown in Figure 52, the first disk on controller vme 1 is unit 0.

Figure 52 ioconfig file with disk numbers assigned



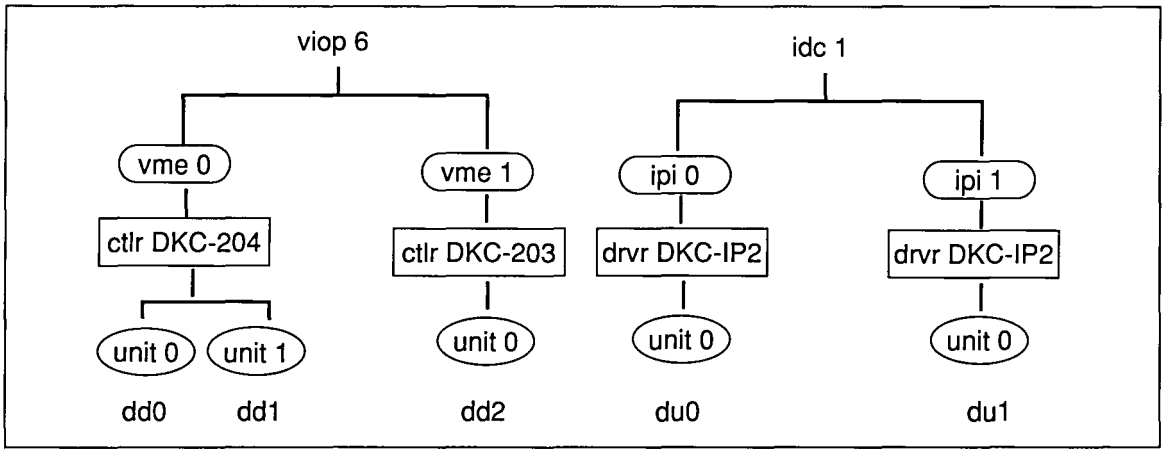

```
# spu -r /ioconfig

viop 6
          vme 0
                    ctlr DKC-204 csr 0x200
                    int 1
                    unit 0 type DKD-206
                    unit 1 type DKD-206
          vme 1
                    ctlr DKC-203 csr 0x000
                    int 1
                    unit 0 type DKD-208
idc 1
          ipi 0
                    drvr DKC-IP2
                    unit 0 type DKD-502
          ipi 1
                    drvr DKC-IP2
                    unit 0 type DKD-502
```

For naming purposes, number disks sequentially starting from 0 for each controller type, starting with the first disk listed in the ioconfig file for each controller type. Name the first Multibus disk listed da0, the next da1, and so forth. Similarly, name the first listed VMEbus disk dd0, the next dd1; the first IDC disk is du0, and the next du1.

Step 3 Add the disk number for each disk to your diagram. Your diagram should now look something like the diagram shown in Figure 53.

Figure 53 Disk configuration diagram with disk numbers assigned



Caution

If you change the order of the entries in the ioconfig file, you must modify the number assigned to the disk. Any changes to the ioconfig file require matching changes to the /etc/fstab file, and to the bootcmd file if the number of the swap device changes.

Step 4 List the file systems that will be part of your environment and the approximate amount of disk space they need. If you plan to use redundant striping, include space for parity or mirroring and hot spare devices.

Step 5 Before deciding on where new file systems will be located, study the existing disk partitioning using the `df` (disk free) command and place this information on your diagram. Enter:

```
% df
```

Figure 54 illustrates output from this command.

Figure 54 Example output from `df` command

```
% df
File system    kbytes    used     avail    capacity   Mounted on
/dev/du0a      45978     27826    13554    67%        /
/dev/dd0h      261215    85615    149478   36%        /doc
/dev/dd0g      401439    218906   142389   61%        /usr
/dev/dd0b      176783    99311    59793    62%        /export
/dev/dd0a      44159     20017    19726    50%        /usr/adm
/dev/du1a      45978     8834     32546    21%        /tmp
/dev/du1b      183402    161746   3314     98%        /sunex
/dev/du1h      275400    235462   12398    95%        /mnt
/dev/dd1a      44159     18294    25865    41%        /usr/spool
```

Note

Only file systems that are mounted appear in `df` command output. Unmounted file systems and swap space do not appear.

From this output, you can determine what partitions are mounted and at what capacity they are being used. For example, in Figure 54, /dev/du1b is being used at 98% capacity, while /dev/dd0h is only being used at 36% capacity.

If any volatile file system exceeds 90%, consider moving it to a larger partition.

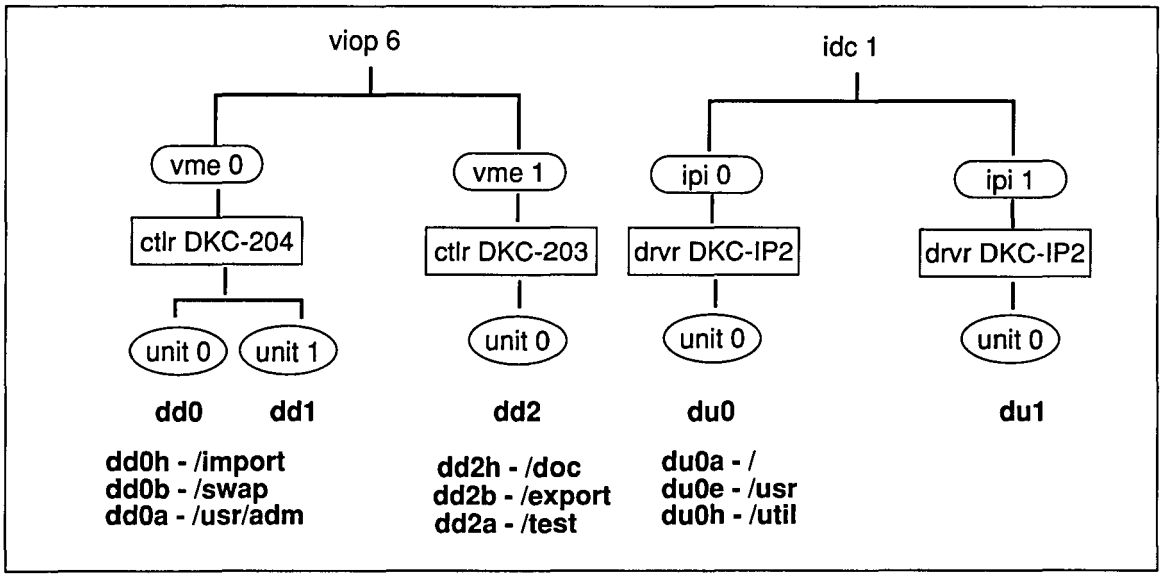
The sum of the used and available kilobytes reported does not add up to the total number of kilobytes on the disk. ConvexOS reserves a percentage of the total number of file system disk free space to prevent severe fragmentation of disks.

Step 6 Place the information from the `df` command output on your diagram.

Step 7 Make your decisions on file system locations and add them to your diagram. Be sure to include the default swap space.

Your diagram should now look something like the diagram shown in Figure 55.

Figure 55 Disk configuration diagram with file system locations



Step 8 Check for existing disk striping and existing hot spares using the `getst` command, and place this information in your diagram.

Stripes are named using the `st#` convention, starting with the number 0 and incrementing by 1 for each new stripe. For example, the first stripe designated is named st0, the second is named st1, and so forth. Check to see whether there are existing stripes by running the `getst` command. Enter:

```
% getst
```

This command gives you information on all existing stripes. If no stripes exist, the system prompt returns with no output from `getst`.

If your system has multiple disk stripes, you can list the stripes for which you want information as arguments to the `getst` command. For example, if your system has two stripes, enter:

```
% getst st0 st1
```

Figure 56 shows example output for this command.

Figure 56 Example output from `getst` command

```
% getst st0 st1
   stripe st0: redundant, sector size 2048 bytes, mounted on /usr/local
       section a: size 49200 Kbytes/partition, blocking factor 8 Kbytes
                  partition 0: du6f (64, 1542) offset 0 Kbytes
                  partition 1: du0f (64, 6) offset 0 Kbytes
                  partition 2: du4a (64, 1025) offset 0 Kbytes
        section b: size 48600 Kbytes/partition, blocking factor 8 Kbytes
                  partition 0: du6f (64, 1542) offset 49200 Kbytes
                  partition 1: du0f (64, 6) offset 49200 Kbytes
   stripe st1: redundant, sector size 2048 bytes, mounted on /bos3
        section a: size 97792 Kbytes/partition, blocking factor 16 Kbytes
                  partition 0: du4f (64, 1030) offset 0 Kbytes
                  partition 1: du2f (64, 518) offset 0 Kbytes
                  partition 2: du1f (64, 262) offset 0 Kbytes
```

From the above output, you can determine that stripe 0 (st0) spans three partitions: partition *f* on IDC disk 6 (du6f), partition *f* on IDC disk 0 (du0f) and partition *a* on IDC disk 4 (du4a); and that stripe 1 (st1) stripes three partitions: partition *f* on IDC disk4 (du4f), partition *f* on IDC disk 2 (du2f) and partition *f* on IDC disk1 (du1f).

Entering `getst` without any options displays information for all existing stripes. `getst` also displays information on any failed devices, including reconstruction status if you are using redundant stripes and hot spares.

To display a list of the available hot spares and their status, enter:

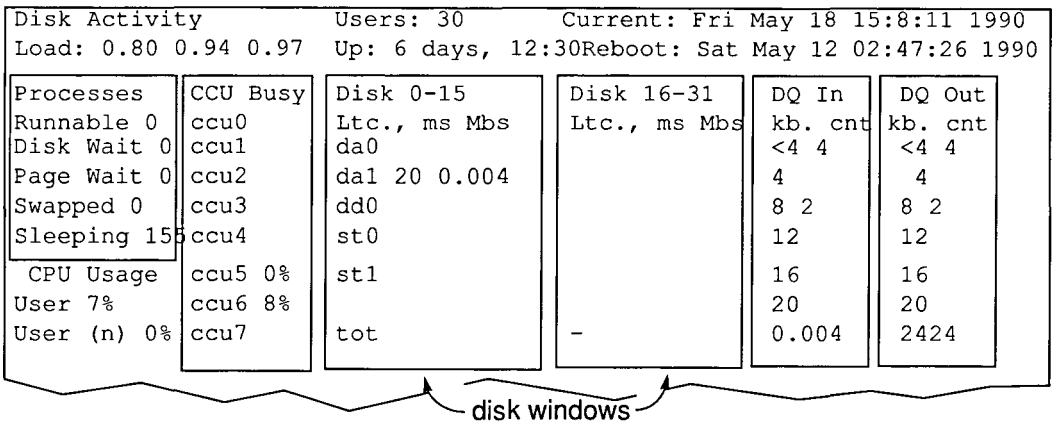
```
% getst -H
```

Step 9 Run `syspic` while the system is running a normal load to check load balance. Use the `-p` disk option to display a picture of the disk statistics:

```
% syspic -p disk
```

Figure 57 illustrates output from this command. If the column marked `ms` in the disk window is greater than 500, that disk is heavily loaded and you should investigate ways to improve load balance. (Stripe Mbs is not added into the total. The total is the sum of all non-striped devices.)

Figure 57 Example `syspic` window



Step 10 Decide on partitions for new file systems and place this information on your diagram.

Assign file systems to specific partitions based on ConvexOS naming conventions, file system space requirements, and performance considerations. A partition can contain only one file system.

There are typically four major file systems to distribute among available disks: / (root), /tmp, /usr, and /mnt. Remember, the root file system includes the /etc, /bin, and /dev directories; the /usr file system includes system programs and other supporting information needed by users; the /mnt file system includes user directories and files; and the /tmp file system contains intermediate files put there by system programs such as compilers, editors, assemblers, and so on. Swap is by default set to the 0b partition.

When establishing the /tmp file system, consider the following information:

- In a configuration with several disks, mount /tmp in partition *a* of the second disk.
- For performance reasons, do not make /tmp a symbolic link to another directory or partition.
- System programs (compilers, editors, assemblers, and so on) create intermediate files in the /tmp directory. Make the file system large enough to accommodate the temporary increases in disk space required by the /tmp directory.
- In a configuration with several disks, use a separate partition for /tmp. This lessens the danger of `fsck` errors in the root partition.

If you plan to use redundant striping and hot spares, keep in mind the criteria for hot spare partitions. To qualify as a hot spare, a partition must have the following qualities:

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

Step 11 Make decisions on disk striping and include this information on your diagram. Decide on:

- Which partitions are striped together
- Whether or not the stripe is redundant
- Whether the redundant stripe is mirrored or parity

Step 12 If you are using parity redundant stripes, you can determine the amount of storage space required for parity information using the `newst` command.

The amount of space required for storing parity information depends on the layout of the stripe. To determine the space requirements for a potential stripe, execute the `newst` command to create the desired stripe and include the `-n` option on the command line. This command generates stripe information without actually creating the stripe. In Figure 58, an example of the `newst` command is shown.

Figure 58 Full output from `newst` command

```
# /etc/newst -nvR st0 du0c dkd-502 du1c dkd-502 du2c dkd-502 du3c dkd-502 \
du4c dkd-502 du5c dkd-502
stripe st0: redundant, sector size 2048 bytes
        section a: size 489296 Kbytes/partition, blocking factor 16 Kbytes
                partition 0: du5c (64, 1283) offset 0 Kbytes
                partition 1: du1c (64, 259) offset 0 Kbytes
                partition 2: du2c (64, 515) offset 0 Kbytes
                partition 3: du3c (64, 771) offset 0 Kbytes
                partition 4: du4c (64, 1027) offset 0 Kbytes
        section b: size 489296 Kbytes/partition, blocking factor 16 Kbytes
                partition 0: du0c (64, 3) offset 0 Kbytes
                partition 1: du1c (64, 259) offset 489296 Kbytes
                partition 2: du2c (64, 515) offset 489296 Kbytes
                partition 3: du3c (64, 771) offset 489296 Kbytes
                partition 4: du4c (64, 1027) offset 489296 Kbytes
        section c: size 489296 Kbytes/partition, blocking factor 16 Kbytes
                partition 0: du0c (64, 3) offset 489296 Kbytes
                partition 1: du5c (64, 1283) offset 489296 Kbytes
/etc/putst /dev/rst0
newst: warning, 'size' & 'cpg' mkfs args are estimates, due to the '-n' option
/etc/mkfs /dev/rst0 2201832 180 7 65536 8192 4 1 32 10 60 2048 60
/etc/fsirand /dev/rst0
# ▮
```

Step 13 To determine parity space requirements, add the kbytes/partition size shown in the output for each section in the stripe. For instance, for sections `a`, `b`, and `c` in the example above, the space used by parity information is 489296 + 489296 + 489296 = 1467888 kbytes. The space available for data in each section is *xxxx*(*n*-1) kbytes, where *xxxx* is the kbytes/partition size for that section and *n* is the number of partitions in the section.

Step 14 Include information for hot spares on your diagram. Hot spares must have the following qualities:

- Sector size less than or equal to that of the stripe device

- Space greater than or equal to that of the partition to be replaced
- Not in the same section as disk it will replace
- Same disk type as the disk it will replace (preferable, but not mandatory)

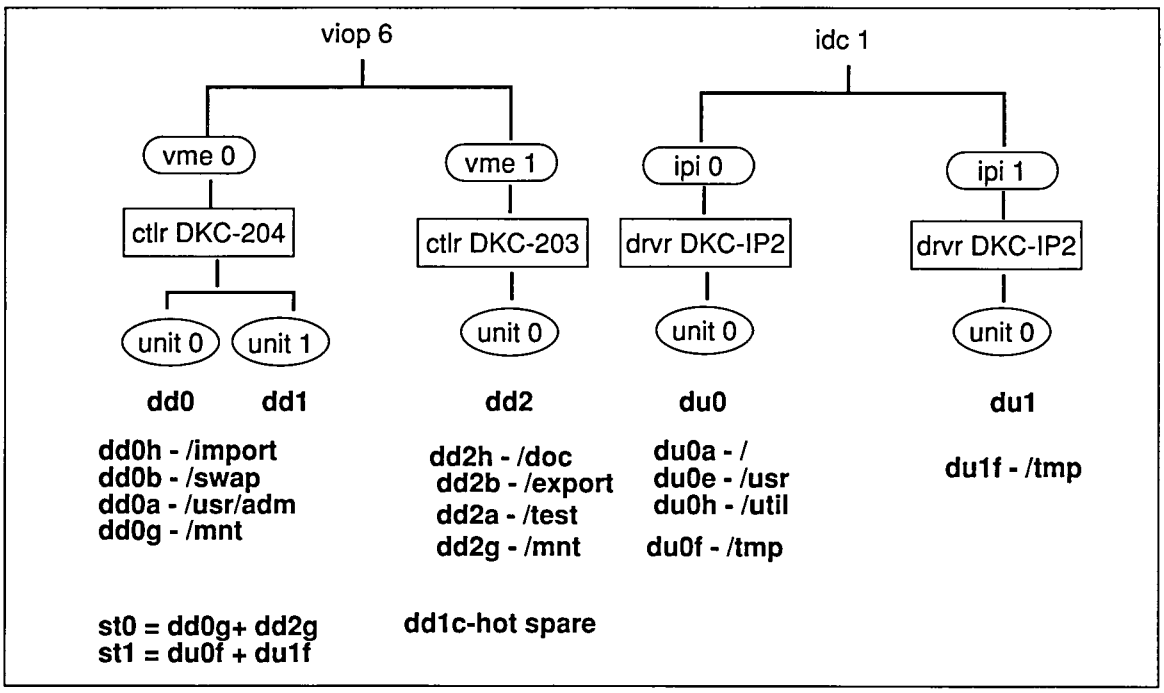
Step 15 Decide on swap space and place this information in your diagram.

When configuring swap space, consider:

- The sum of available real memory plus swap space determines the size of the largest set of processes that can run on your system at the same time.
- System memory requires about 3 Mbytes plus 10 percent of physical memory; the rest is available for user programs.
- To run with reasonable throughput, the total amount of swap space plus 75% of available memory must be greater than the sum of all process virtual sizes.

Your diagram, like the one shown in Figure 59, should now contain all information you require to create disk partitions in your environment.

Figure 59 Disk configuration diagram with partition and stripe information

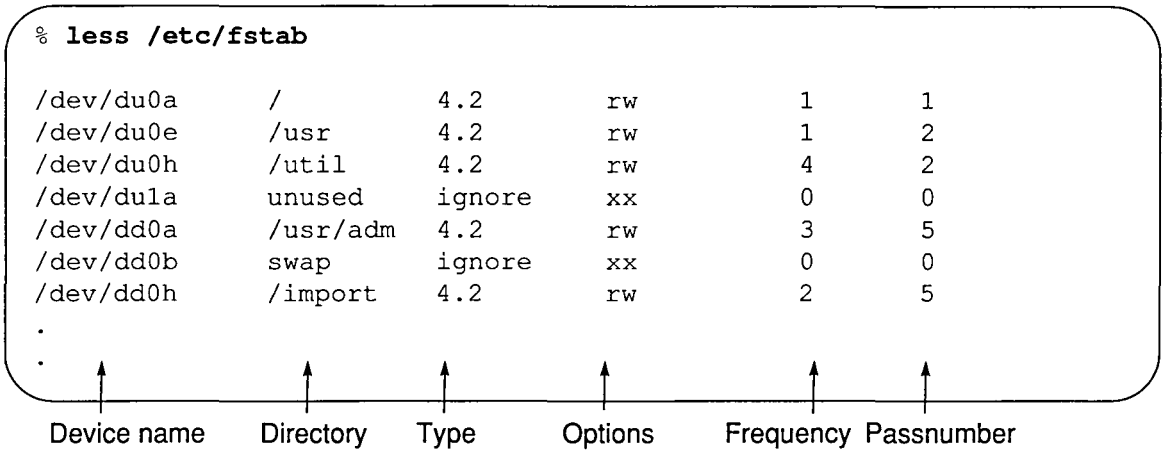


Step 16 Look at the /etc/fstab file to be sure selected partitions do not conflict with existing partitions and make any necessary adjustments to your diagram. Enter

```
% less /etc/fstab
```

The fstab file describes all file systems, whether they are mounted or unmounted, as well as the swap partitions. Figure 60 illustrates an example of the contents of the fstab file.

Figure 60 Example /etc/fstab file



In the fstab file, the directory is the name of the directory where the file system is mounted.

Type describes the file system. This can be:

`4.2`	BSD type file system.
`nfs`	NFS file system.
`swap`	Swap partition.
`ignore`	Indicates the system should ignore this record because the partition is not used. This should be specified for the default swap partition, normally dx0b. The kernel automatically mounts the default swap partition, so specifying swap results in an invalid argument error at boot time. You should also specify `ignore` for any hot spare devices you add to /etc/fstab.

Options can be any number of the following values, separated by commas:

`rw`	Permits reads and writes to the files (default).
`ro`	Only permits files in the file system to be read.

`suid` Setuid execution is permitted (default).

`nosuid` Setuid execution is not permitted.

`quota` Disk usage limits are enforced if quotas have been set up. See Chapter 9,"Setting quotas on disk space use," for more details.

`noquota` Disk usage limits are not enforced (default).

`nolf` New large files (files greater than 2 gigabytes in size) are prohibited in this file system. An attempt to place a large file in this file system may truncate the file.

Specify `nolf` for any file systems where you do not want large files. For more information on large files, see "Large files" in the *ConvexOS Extensions User's Guide.*

`hide` Do not mount this file system with execution of the `mount -a` command.

The frequency field indicates how often file systems should be backed up to tape. This can be one of the following:

0 Never back up the file system.

1 Back up the file system daily (this is the recommended frequency).

2 Back up the file system every two days.

Note

The frequency field is only used as a tool to know how often to dump the file system. It does not cause the file system to be dumped automatically.

Passnumber is used by the `fsck` utility to determine on which pass to check a file system. It uses the information here to inspect groups of disks in parallel. For instance, all file systems with a 2 in passnumber are checked on the second pass of `fsck`. The `preen` utility does not use this field.

0 A file system with this number is not checked. For example, file systems set aside for swap space should have this field set to a 0.

1 A file system with this number is checked on the first pass. The root file system should be checked on the first pass.

2 A file system with this number is checked on the second pass. All file systems on *a* partitions should be checked on the second pass.

3 All file systems on *d* partitions should be checked on the third pass.

4 All file systems on *e* partitions should be checked on the fourth pass.

5 Large user file systems should be checked on the fifth (last) pass.

See the fstab(5) man page for more details on the fstab file.

Step 17 Decide on block and fragment size.

Block and fragment sizes affect system performance. You can specify the block and fragment size for each file system when creating the file system. Otherwise, the default specified in the /etc/disktab file for blocks and fragment sizes are used. See the disktab(5) man page for more details.

Note

For redundant stripes or any stripes containing IDC disks, the minimum fragment size is 2 kbytes.

Theoretically, small fragment size and large block size yield the best performance and greatest space efficiency. Files in large blocks take fewer disk operations and transfer faster, so a large block size yields better performance.

But, because larger blocks yield larger fragments and even a one-character file uses an entire fragment on disk, disk space efficiency can suffer, especially in a file system with many small files. Therefore, when you increase transfer speed, space efficiency suffers. Using disk space more efficiently for small files means slower transfers.

File system block size affects the time required to read and write data for a file in the buffer cache. The block and fragment sizes you choose for a file system depend primarily on the size of files in the file system. If your users have many small files, limit the minimum space used for a single file to minimize wasted disk space. If your users have large files or need maximum performance, maximize the amount of data that can be transferred at once. The following examples show how disk space requirements for various file systems are affected by varying the block and fragment sizes.

Example 1: File system /mnt contains a mix of user files, mostly program source files and executables. In this example, the recommended block size is 16 kbytes and fragment size is 2 kbytes, although it means an 8% loss of space (on Multibus and VMEbus disks) because the fragment size is 2 kbytes rather than 1 kbyte, as shown in Table 12.

Table 12 Recommended block and fragment sizes

Block size	Fragment size	Disk space used: /mnt
4 kbytes	512 bytes	245 Mbytes
8 kbytes	1 kbyte	251 Mbytes
16 kbytes	2 kbytes	264 Mbytes*
32 kbytes	4 kbytes	294 Mbytes
64 kbytes	8 kbytes	362 Mbytes
*Recommended size. System swith limited disk space can select the next smaller block and fragment size to preserve as much disk space as possible. IDC disks have a minimum fragment size of 2 kbytes.		

Example 2: File system /work1 consists almost entirely of small source and data files. Most files are less than 1 kbyte, so each increase in fragment size results in a large increase in file system waste. The recommended block size is 8 kbytes and fragment size is 1 kbyte, resulting in an 11% loss of space but higher throughput, as shown in Table 13.

Table 13 Recommended block and fragment sizes

Block size	Fragment size	Disk space used: /work1
4 kbyes	512 bytes	66 Mbytes
8 kbytes	1 kbyte	73 Mbytes*
16 kbytes	2 kbytes	87 Mbytes
32 kbytes	4 kbytes	118 Mbytes
64 kbytes	8 kbytes	185 Mbytes
*Recommended size. System swith limited disk space can select the next smaller block and fragment size to preserve as much disk space as possible. IDC disks have a minimum fragment size of 2 kbytes.		

Example 3: File system /work2 consists primarily of data files greater than 1 Mbyte. The recommended block size is 64 kbytes and the fragment size is 8 kbytes, wasting only 5% of disk space and resulting in higher performance, as shown in Table 14.

Table 14 Recommended block and fragment sizes

Block size	Frgment size	Disk space used: /work2
4 kbytes	512 kbytes	590 Mbytes
8 kbytes	1 kbyte	592 Mbytes
16 kbytes	2 kbytes	596 Mbytes
32 kbytes	4 kbytes	603 Mbytes
64 kbytes	8 kbytes	622 Mbytes*
*Recommended size. System swith limited disk space can select the next smaller block and fragment size to preserve as much disk space as possible. IDC disks have a minimum fragment size of 2 kbytes.		

See Chapter 8, "Checking the file system," in the *Operations Guide* for more details on block sizes.

Step 18 Decide on inode count for each file system.

The structure of ConvexOS file systems requires one inode per file in the file system. Each inode uses 128 bytes on the disk. Inodes take up space; do not waste space by having more inodes than necessary for a file system. Because one inode is needed per file, a file system with many small files requires more inodes than a file system with a few large files.

Note

Having enough inodes is important. Even if the files on a file system are only taking up half the available disk space, you cannot create new files if there are no free inodes.

The number of inodes should range from a minimum of one inode per 16 kbytes of storage to a maximum of one inode per 4 kbytes of storage. The number of inodes in a file system is established when creating a new file system using the `newfs` or `newst` command. The default number of inodes is based on one inode per 2 kbytes of data space.

You can use the `df` command to help determine whether or not the inodes are proportionately effective in your existing file systems. This can help you in making future decisions. Run `df` to check the current file system capacity. Enter:

```
% df
```

Figure 61 illustrates sample output from this command.

Figure 61 `df` sample output

```
% df
File system    kbytes  used     avail    capacity    Mounted on
/dev/du0a      45978   27826    13554    67%         /
/dev/du0e      275970  238936   9436     96%         /usr
/dev/du0h      275400  216042   31818    87%         /util
```

Then run `df` with the `-i` option to check the number of free inodes and the number of inodes used on existing file systems:

```
% df -i
```

Figure 62 illustrates example output from this command.

Figure 62 `df -i` sample output

```
% df -i
File system    iused   ifree    %used    blks/frags  Mounted on
/dev/du0a      1451    4693     24%      16k/2k      /
/dev/du0e      9607    19065    34%      16k/2k      /usr
/dev/du0h      12024   16648    42%      16k/2k      /util
```

If a file system is running at a high capacity, yet there are many free inodes, you have probably specified too many inodes for the file system.

Configuring disk partitions

Now that you have considered all the necessary information and formulated the plan for your disk system, use the following procedure to implement the plan developed in the previous section. The procedure is broken into five sections:

- Preparing the fstab file and making the devices
- Configuring single disk partitions
- Configuring striped partitions
- Integrating disk system changes

Begin with the steps in the "Preparing..." section. Then, depending on the types of partitions you need to add, perform the steps in one or more of the "Configuring..." sections. After you complete as many of the configuration procedures as you need, perform the steps in the "Integrating disk system changes" section.

Some of the steps require you to modify files located on the SPU using an editor. If you do not want to use the `xed` editor available on the SPU, and you are not changing the partition that is mounted on /usr, you can mount /usr. This is where the `vi` and `emacs` editors reside. Once you mount /usr, you can use the `vi` or `emacs` editor to modify the SPU files.

Preparing the fstab file and making the devices

Step 1 Log in as the superuser.

Step 2 Make a back up copy of the existing /etc/fstab file. Enter:

```
# cp /etc/fstab /etc/fstab.old
```

The fstab file contains file system tables that specify disk partitions on which the file systems are to be mounted. Several utilities such as `fsck`, `mount`, `umount`, and `dump` use information in the fstab file. You must modify this file to reflect your decisions. An example /etc/fstab file is shown in Figure 63.

Figure 63 Example /etc/fstab file

```
/dev/du0a       /          4.2       rw        1        1
/dev/du0e       /usr       4.2       rw        1        2
/dev/da0g       /util      4.2       rw        4        2
/dev/dd0b       swap       swap      xx        0        0
```

Device name | Directory | Type | Options | Frequency | Passnumber

Step 3 Edit the /etc/fstab file so that the system recognizes the new partitions, partition striping, and swap partitions. See Figure 60 for an example /etc/fstab file.

Each partition should be represented with a line that includes information in the format:

dev_name dir type opts freq passno

where

dev_name is the device name.

dir is the name of the directory on which the file system is to be mounted.

type is the type of file system. This can be:

`4.2` BSD 4.2 file system.

`nfs` NFS file system.

`swap` Swap partition.

`ignore` Ignore this record. This should be specified for the default swap partition, normally da0b. The kernel automatically mounts the default swap partition, so specifying swap results in an invalid argument error at boot time. Also specify ignore for any hot spare devices.

opts specifies how the file system can be used. This can be any number of the following values separated by commas:

`rw` Permits read and writes to the files (default).

`ro` Only permits files in the file system to be read.

`suid` Setuid execution permitted (default).

`nosuid` Setuid execution not permitted.

`quota` Disk usage limits are enforced if quotas have been set up. Refer to Chapter 9, "Setting quotas on disk space use," on page 191, for more details on setting up disk quotas.

`noquota` Disk usage limits are not enforced (default).

`nolf` New large files (files greater than 2 gigabytes in size) are prohibited in this file system. An attempt to place a large file in this file system may truncate the file.

Specify `nolf` for any file systems where you do not want large files. For more information on

large files, see "Large files" in the *ConvexOS Extensions User's Guide*.

`hide` Do not mount this file system with execution of the `mount -a` command.

freq is how often file systems should be backed up to tape:

- `0` Never back up the file system.
- `1` Back up the file system daily (this is the recommended frequency).
- `2` Back up the file system every two days.

and so forth.

Note

The *freq* field is only used as a tool to know how often to dump the file system. It does not cause the file system to be dumped automatically.

passno is used by the `fsck` utility to determine on which pass to check a file system. It uses the information here to inspect groups of disks in parallel. For instance, all file systems with a `2` in passnumber are checked on the second pass of `fsck`.

- `0` A file system with this number is not checked. (For example, file systems set aside for swap space should be set to a `0`.)
- `1` A file system with this number is checked on the first pass. The root file system should be checked on the first pass.
- `2` A file system with this number is checked on the second pass. All *a* partitions should be checked on the second pass.
- `3` All file systems on *d* partitions should be checked on the third pass.
- `4` All file systems on *e* partitions should be checked on the fourth pass.
- `5` Large user file systems should be checked on the fifth (last) pass.

Step 4 Change the working directory to the /dev directory. Enter:

```
# cd /dev
```

Step 5 Create block and raw device files in the /dev directory for each new partition and new disk stripe using the /dev/MAKEDEV command. Do not do this for existing stripes or partitions. For example, to add a new disk named du3 and two stripes to your disk system, enter:

```
# /dev/MAKEDEV st0 st1 du3
```

This creates raw (rst0, rst1, rdu3a-rdu3h), block (st0, st1, du3a-du3h) and control device entries (rst0-ctl, rst1-ctl, rdu3-ctl) in the /dev directory for both stripes and the partition. For more details, see Chapter 2, "Adding devices."

Configuring single disk partitions

Follow these steps for each file system to be created on a single disk partition:

Step 1 If you are not logged in as superuser, do so now.

Step 2 Use the `qst` command to check that the partition you are creating does not conflict with existing striped partitions. The `qst` command has the syntax:

`qst` *disk_device*

disk_device is the name of the disk device without the partition letter.

For example, if you are creating the da0g partition, enter the following:

```
# qst da0
```

If, in the output, there is any mentioning of conflicting striped partitions, which, in this case, are da0c, da0d, da0e, da0f, or da0g, you must plan a different scheme for configuring your current partitions.

Step 3 Use the `mount` command with the following format to ensure that the device you are going to use is not already in use with another file system:

`mount | grep` *disk_device*

where *disk_device* is the name of the disk device without the partition letter.

For example, if you are creating the da0g partition, enter the following:

```
# mount | grep da0
```

If, in the output, there is any mentioning of conflicting partitions, which, in this case, is da0c, da0d, da0e, da0f, or da0g, you must plan a different scheme for configuring your current partitions.

Step 4 Use the `grep` command with the following format to ensure that the device you are going to use does not have a conflicting entry in the /etc/fstab file:

`grep` *disk_device* `/etc/fstab`

disk_device is the name of the disk device without the partition letter.

For example, if you are creating the da0g partition, enter the following:

```
# grep da0 /etc/fstab
```

If, in the output, there is any mentioning of conflicting partitions, which, in this case, is da0c, da0d, da0e, or da0f, you must plan a different scheme for configuring your current partitions.

Step 5 Back up with level 0 dumps all files located on the disk you are going to partition. See the *ConvexOS Tape System Operator's Guide* for more details on performing backups.

Step 6 Find the `newfs` command options that match the parameters you wish to define for the new file system. Some of the more common options are listed below. Refer to the newfs(8) man page for a comprehensive list.

`-b` *block_size*	Specifies the block size in bytes. For example, you can specify either 64k or 65536. See Table 11 on page 66 for specific information on block and fragment sizes.
`-f` *frag_size*	Specifies the fragment size in bytes. For example, you can specify 8K or 8192. See Table 11 on page 66 for specific information on block and fragment sizes.
`-m` *free_space*	Specifies the percentage of space reserved from normal users. The default value is 10.
`-n`	Displays the parameters that would apply to the new file system, but does not create the file system.
`-I` *inode #*	Specifies the number of inodes in the file system.

For instance, if your disk plan includes specific block and fragment sizes, select the `-b` and `-f` options to specify those values. If you do not use these options, the system uses default values specified in the /etc/disktab file. (See Figure 64 for an example /etc/disktab file.)

Caution

The `newfs` command destroys any data that currently exists on the file system, so be sure to verify the device name before executing this command.

Step 7 Issue the `newfs` command with the appropriate options to create the file system. To test the command before actually executing it, direct the output to the screen using the `-n` option.

The format for the `newfs` command is:

`newfs` *options raw_device disktype*

options — Specify parameters for the new file system. The system uses values from /etc/disktab for any values not specified on the command line.

raw_device — Specifies the device partition the file system will use. Always designate the raw device name, not the block device name. For example, if you create a file system on da0c, you should designate rda0c. You can specify only one raw device for a single partition.

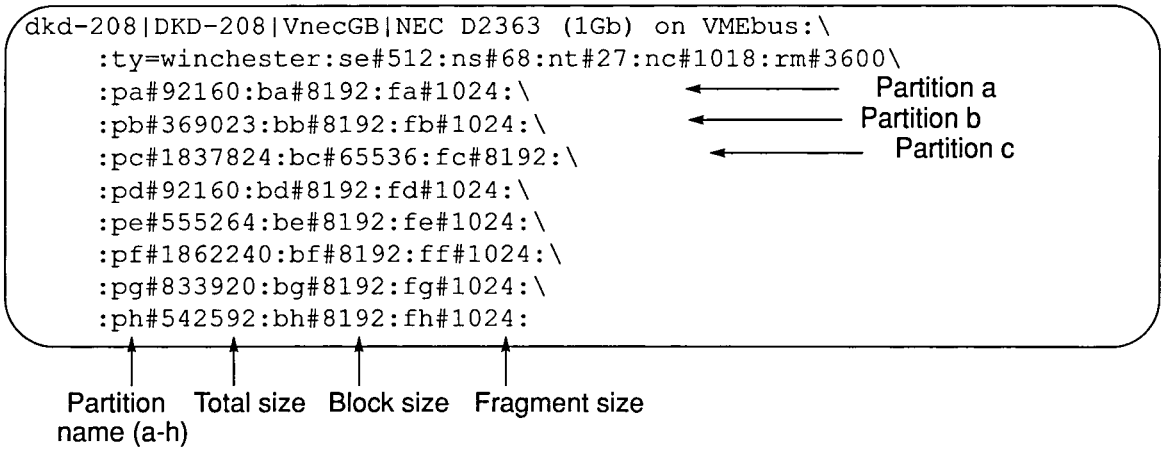
disktype — Specifies the type of disk on which the file system is created. The device name should match one of the entries in the /etc/disktab file. For example, you could specify dkd-208, DKD-208, or VnecGB if your disk type was the type shown in Figure 64.

For example, the following command creates a new file system on partition *c* of disk da0 of type dkd-005, with a block size of 64 kbytes and a fragment size of 8 kbytes:

```
# newfs -b 64k -f 8k /dev/rda0c dkd-005
```

When creating the new file system, the system uses the information stored in the /etc/disktab file for disk geometry and file system partition information, such as the default values for block and fragment sizes. You can override these defaults using options provided with this command. Never change the values in the /etc/disktab file, as this would badly confuse several file system utilities and could result in a panic. Figure 64 shows a sample /etc/disktab file.

Figure 64 Example /etc/disktab file



Configuring striped partitions

Follow these steps for each file system to be created on a striped disk partition:

Step 1 If you are not logged in as superuser, do so now.

Step 2 Use the `qst` command to check that the striped partition you are creating does not conflict with existing partitions. The `qst` command has the syntax:

`qst` *disk_device*

disk_device is the name of the disk device without the partition letter.

For example, if you are creating the da0g partition, enter the following:

```
# qst da0
```

If, in the output, there is any mentioning of conflicting partitions, which, in this case, are da0c, da0d, da0e, da0f, or da0g, you must plan a different scheme for configuring your current stripe partitions.

Step 3 Use the `mount` command with the following format to ensure that the device you are going to use is not already in use with another file system:

`mount | grep` *disk_device*

disk_device is the name of the disk device without the partition letter.

For example, if you are creating the da0g partition, enter the following:

```
# mount | grep da0
```

If, in the output, there is any mentioning of conflicting partitions, which, in this case, is da0c, da0d, da0e, da0f, or da0g, you must plan a different scheme for configuring your current stripe partitions.

Step 4 Use the `grep` command with the following format to ensure that the device you are going to use does not have a conflicting entry in the /etc/fstab file:

`grep` *disk_device* `/etc/fstab`

disk_device is the name of the disk device without the partition letter.

For example, if you are creating the da0g partition, enter the following:

```
# grep da0 /etc/fstab
```

If, in the output, there is any mentioning of conflicting partitions, which, in this case, is da0c, da0d, da0e, or da0f, you must plan a different scheme for configuring your current stripe partitions.

Step 5 Back up with level 0 dumps all files located on the disk you are going to partition. See the *ConvexOS Tape System Operator's Guide* for more details on performing backups.

Step 6 Change to the root directory:

```
# cd /
```

Step 7 Unmount any existing, non-striped file systems that you plan to stripe. For example,

```
# umount /mnt
```

unmounts the /mnt file system.

Caution

This command destroys any data that currently exists on the file system, so be sure to verify the device name before executing this command.

Step 8 The `newst` command creates a description of the stripe, loads it into the kernel, and creates a file system on the stripe partition. Find the `newst` command options that match the parameters you wish to define for the new file system.

For instance, if your disk plan includes specific block and fragment sizes, select the `-b` and `-f` options to specify those values. If you do not use these options, the system uses default values specified in the /etc/disktab file.

Some of the more common options are listed below. Refer to the newst(8) man page for a comprehensive list.

`-b` *block_size* — Specifies the desired block size in bytes. For example, you can specify either 64k or 65536. See Table 11 on page 66 for specific information on block and fragment sizes.

`-f` *frag_size* — Specifies the desired fragment size in bytes. For example, you can specify either 8 kbytes or 8192. See Table 11 on page 66 for specific information on block and fragment sizes.

`-I` *inode #* — Specifies the number of inodes in the file system.

`-P` *partition #* — Specifies the number of disk partitions in a stripe section. If you specify a number of partitions smaller than the number of disk partitions in the stripe, the extra partitions are stacked into a second stripe section. Stripes with a smaller number of partitions have a longer mean time to failure.

`-S` — The number of spare physical sectors per cylinder.

The following option is valid only for redundant stripes:

`-R` — Enables redundant striping. The `-R` option alone establishes parity on the file system if the stripe contains more than two disk partitions. A redundant stripe of only two partitions is always mirrored. To force mirroring of a stripe containing more than two disk sections, specify `-P2` in addition to the `-R` option. (For descriptions of parity and mirroring, see the "Redundant stripe partitions" section earlier in this chapter.)

Step 9 Issue the `newst` command with the appropriate options to create the file system. To test the command before actually executing it, direct the output to the screen using the `-n` option.

Caution

The *newst* command destroys any data that currently exists on the file system, so be sure to verify the device name before executing this command.

The format to configure a stripe with the `newst` command is:

`newst` *options st_dev diskdev1 type1 [diskdev2 type2...]*

which has the following components:

options Specify parameters for the new file system. If you do not use these options, the system uses the default values specified in the /etc/disktab file.

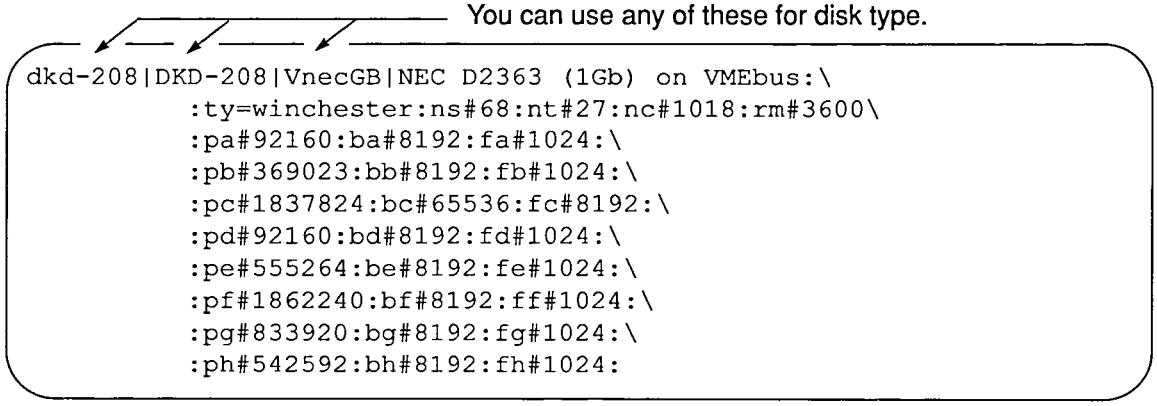
st_dev Specifies the name of the striped device the file system will use.

diskdev Specifies the name of the first disk partition included in the stripe. You can specify multiple device and type pairs for a single stripe.

type Specifies the type of disk for the first partition the file system is being created on. This name should match one of the entries in the /etc/disktab file. For example, you could specify dkd-208, DKD-208, or VnecGB if your disk type was the type shown in Figure 64. You can specify multiple device and type pairs for a single stripe.

When creating the new file system, the system uses the information stored in the /etc/disktab file for disk geometry and file system partition information, such as the default values for block and fragment sizes. You can override these defaults using options provided with this command. Never change the values in the /etc/disktab file. Figure 65 shows a sample /etc/disktab file.

Figure 65 Example /etc/disktab file



The following command creates a new striped file system (st1) on partition *g* of disk da0 of type dkd-001 and partition *g* of disk da1 of disk type dkd-001, with a block size of 64 kbytes and a fragment size of 8 kbytes:

```
# newst -b 64k -f 8k /dev/rst1 /dev/rda0g dkd-001 /dev/rda1g dkd-001
```

The command

```
# newst -R /dev/rst1 /dev/rda0g dkd-001 /dev/rda1g dkd-001 /dev/rda2g dkd-001
```

creates a one-section, three disk-wide redundant stripe.

Hot spare partitions

If you configured redundant stripes, you may specify hot spare partitions for those stripes. To configure a hot spare partition, use the following form of the `newst` command:

`newst -H options` [*st_dev...*] *diskdev1 type1*

This form uses the following option:

`-H` Adds the specified disk to the hot spare list for redundant striped devices. In order to qualify as a hot spare for a stripe device, a hot spare must have a sector size less than or equal to the stripe sector size. The spare must also have sufficient available space.

The command

```
# newst -H /dev/rst1 du3c dkd-001
```

adds disk du3, partition c, to the hot spare list. The inclusion of /dev/rst1 in the command string gives hot spare du3c an affinity for stripe rst1. A hot spare with an affinity for a particular stripe will be chosen over other hot spares to replace a failed disk partition in that stripe. A hot spare with affinities specified may only replace a disk in the stripes for which it has an affinity.

Enabling disk system changes

After carefully planning your disk system, use the following procedure to enable changes.

Step 1 Change to single-user mode by issuing the `shutdown` command.

You should partition disks in single-user mode to ensure user-data integrity. If there are no other users logged on the system, you can issue the command:

```
# shutdown now "to reconfigure disks"
```

The system immediately shuts down to single-user mode with no warning. If there are other users logged on the system, you can specify the number of minutes to wait until shutdown. The following command waits 10 minutes before shutdown and periodically sends messages to users informing them of the impending shutdown:

```
# shutdown +10 "to reconfigure disks"
```

Caution

Executing the commands required to create file systems or to stripe partitions destroys all data previously stored on the partitions.

Step 2 Back up with level 0 dumps all files located on the disk you are going to partition. See the chapter, "Performing backups and restoring files," in the *ConvexOS Tape System Operator's Guide* for more details on performing backups.

Step 3 Change your working directory to the root directory. Enter:

```
# cd /
```

Step 4 Create a mount point for each new file system. Each newly created file system (device) must have a directory on which it is mounted. The directory name should match the name given to the file system in the /etc/fstab file in Step 7 of the "Preparing" procedure. For example, if design is a new file system, create a top-level directory in the root directory called design. Enter:

```
# mkdir /design
```

Step 5 Change the permissions on this directory to allow access to the mount point. Enter:

```
# chmod 777 /design
```

Step 6 Mount the desired file systems. File systems must be mounted before they can be accessed. For example, to mount the /design file system, enter:

```
# mount /design
```

If you want to mount all disk partitions specified in the /etc/fstab file, use the `-a` option. For more information on the `mount` command, see the mount(8) man page.

Step 7 Use the `df` command to check mounted file systems. Enter:

```
# df -i
```

In the `df` output, check the inode count, block and fragment size, and whether or not the file system is mounted. If the file system does not appear on this output, either it is not mounted or it is hidden. To check a specific file system, include the file system name you want to check as an argument to the `df` command.

Step 8 If `update` is not already running, use `update` to flush the disk buffers. Enter:

```
# update &
```

This command flushes the disk buffers at 30-second intervals. This step is in preparation for restoring files from back-up tapes to disk. Restoring files to disk from back-up tapes causes major changes to the file system. This command minimizes the chances of a disk becoming seriously corrupted if a system crash occurs during the restoration process.

Step 9 Restore the files backed up on tape. See the *ConvexOS Tape System Operator's Guide* for information on backing up and restoring files.

Step 10 Unmount the file systems. This step is in preparation for running an integrity check on the disks. If you only added a couple of file systems and do not want to check the integrity for previously existing file systems, you can unmount only the file systems you just added. For example, to unmount the /design file system, enter:

```
# umount /design
```

If you want to unmount all disk partitions specified in the /etc/fstab file, use the `-a` option.

Step 11 Check disk integrity using either the `preen` command or the `fsck` command.

Use the `fsck` command if you have added only a couple of file systems and do not want to check the integrity for previously existing file systems. For example, to run a check on the da5c file system, enter:

```
# fsck /dev/rda5c
```

The `fsck` utility tests the internal structure of a file system, ensuring that each inode has the appropriate number of disk blocks assigned, the reference count matches the appropriate number of path names, the list of free disk blocks is accurate, and the header information is accurate.

Use the `preen` command if you have added many new file systems and do not wish to check them individually. Make sure all file systems are unmounted, then enter:

```
# preen -f
```

The `preen` command runs parallel `fsck` checks on all the file systems specified in the /etc/fstab file. If an error occurs while `preen` is running, you receive the following message on your console:

```
RUN fsck MANUALLY
```

The file system that requires a manual `fsck` check is also designated. If this happens, start an `fsck` check on the specified file system. For example, if you receive this message for the da5c file system, enter:

```
# fsck /dev/rda5c
```

Note that you enter the raw-device name of the file system to check.

See the fsck(8) man page or the *Operations Guide*, "Checking the file system," for more details on the `fsck` command.

Step 12 Mount the desired file systems. For example, to mount the /design file system, enter:

```
# mount /design
```

If you want to mount all disk partitions specified in the /etc/fstab file, use the `-a` option.

Step 13 Set disk quotas as described in Chapter 9, "Setting quotas on disk space use."

Step 14 Boot to multiuser mode by pressing **CTRL-d**. This brings up the system header information and the login prompt. This can take a few minutes.

Update your back-up scripts to recognize the new partitions. See *Managing ConvexOS: Operations Guide*, "Performing backups and restoring files," for more details on updating backup scripts.

Configuring swap space

This section describes how to configure and enable swap space on your system. Follow this procedure after you have planned your disk system from the section, "Planning your disk system," on page 82.

Step 1 Edit the /mnt/os/bootcmd.local file located on the SPU. The partitions specified in this file override those in the /mnt/os/bootcmd file (usually da0b, dd0b, or du0b). The first swap partition specified in the bootcmd.local file becomes the default swap partition.

If there is no bootcmd.local file currently on your system, skip to Step 3.

If the file does exist, make a back-up copy before editing it. Enter at the root prompt:

```
# spucmd cp /mnt/os/bootcmd.local /mnt/os/bootcmd.local.old
```

Step 2 Copy the file to the /tmp directory. Enter at the root prompt:

```
# spu -r /mnt/os/bootcmd.local > /tmp/bootcmd.local
```

Step 3 Add or delete new swap entries to the /tmp/bootcmd.local file using a standard text editor. (Create the file if it does not already exist.) Use the format:

`swap on` *partition*`[,` *partition*`...]`

partition is a swap partition. If adding more than one partition as swap, separate each partition with a comma.

For example, to designate partition dd0b, dd3c, and dd4c as swap partitions, enter the following line in the bootcmd.local file:

```
swap on dd0b, dd3c, dd4c
```

You should only have one "swap on" line in your bootcmd.local file.

Step 4 Copy the modified file to the SPU. Enter at the root prompt:

```
# spu -w /mnt/os/bootcmd.local < /tmp/bootcmd.local
```

Step 5 In the /etc/fstab file, label all swap partitions "swap," except for the default swap partition, which is the first entry and should be marked "ignore." Following is an example of swap space defined in the /etc/fstab file:

```
/dev/dd0b swap_1of3 ignore rw 0 0
/dev/dd3c swap_2of3 swap rw 0 0
/dev/dd4c swap_3of3 swap rw 0 0
```

Step 6 Shut down to the SPU. Enter at the root prompt:

```
# shutdown -h now
```

Step 7 Boot to single-user mode to test the new kernel for swap space recognition. From the SPU prompt, enter

```
spu> boot single
```

Step 8 If you receive any errors, for example

```
/dev/da1g invalid device
```

check:

- That the default swap partition, or the first one specified in the bootcmd.local file, says `ignore` instead of `swap` in the /etc/fstab file.
- That the mnt/os/bootcmd.local file on the SPU is properly set up.

Shut down to the SPU and boot to single-user mode again if necessary.

Step 9 When you no longer recieve error messages, shut down to the SPU for a final time with the following command:

```
# shutdown -h now
```

Step 10 Boot to multiuser mode from the SPU with the following command:

```
spu> boot
```

It is advisable to shut down to the SPU and boot to multiuser from there on the last boot procdure. In the event that your system comes down unexpectedly—for example, due to a power outage—it comes up after an automatic reboot to the level of the last boot. In most cases, you will want your system to boot to multiuser mode during an automatic reboot.

Scheduling file system backups

4

Data stored on disks can be lost due to hardware or software problems with the disk, damage to equipment, or accidental deletion of files. Making back-up tapes of the information on disk on a regular basis ensures against loss of data by allowing you to recover files that are lost or corrupted.

This chapter discusses how to plan for scheduled backups. For information and procedures on how to back up and restore files, refer to the *ConvexOS Tape System Operator's Guide* (DSW-397).

Overview of backing up files

The back-up process copies files from disks to back-up tapes, also known as dump tapes. You can perform either a full dump or incremental dump:

- A full dump backs up all the files in a file system.
- An incremental dump backs up only those files that have changed since the last full or incremental dump, whichever is most recent. You can recover either individual files or an entire file system from these tapes.

There are a number of utilities that you can use to dump file systems to tape: `dump`, `xdump`, and `rdump`.

The `dump` and `xdump` utilities back up files from a local machine; `rdump` dumps files over an Ethernet. The `rdump` utility is part of the ConvexOS Internet Services, which is an optional product that will not exist on your machine unless you have installed it.

The `dump` and `xdump` utilities are identical in function. However, `xdump` runs from two to ten times faster than `dump` by using shared memory, asynchronous I/O, and faster disk-reading algorithms. If you are using ANSI-labeled tapes, you are not able to use `xdump`, because asynchronous I/O is not supported on labeled tapes.

In the remainder of this chapter, references to the `dump` utility apply equally to `dump`, `xdump`, and `rdump`.

The `dump` utility stores files on tape with path names excluding the name of the mounted file system to which they belong. For example, a dump of the /mnt file system stores the file named /mnt/smith/file1 as ./smith/file1. Complex mount-point names can be more confusing. For example, if a file system is mounted on /usr/external, a dump of the /usr/external file system stores the file named /usr/external/smith/file1 as ./smith/file1.

The `dump` utilities examine directories within a file system to select the files to be dumped based on the dump levels recorded in the /etc/dumpdates file. The /etc/dumpdates file contains a list of the dates on which file systems were dumped and the dump level. An example /etc/dumpdates file is shown in Figure 66. The `dump` utility only dumps files modified after the last date of a lower level dump.

Figure 66 An example /etc/dumpdates file

```
/dev/rst3 0 Fri Feb 20 12:40:11 1990
/dev/rst3 1 Sat Feb 21 12:46:52 1990
/dev/rst4 0 Sat Feb 21 14:19:16 1990
/dev/rst3 1 Mon Feb 23 08:40:10 1990
```

If a system is active, that is, running jobs, timesharing, or batch processing, files can change while the dump is executing. If this is the case, files copied to tape may not be copied accurately. This could cause problems when restoring the files.

To ensure accurate backups, run the system in single-user mode during backups, or unmount the file systems being dumped before dumping.

If it is absolutely necessary to backup file systems while the system is active, take an incremental dump immediately after the dump saves the file. This increases your chances of accurately capturing the contents of the file system.

Planning backups

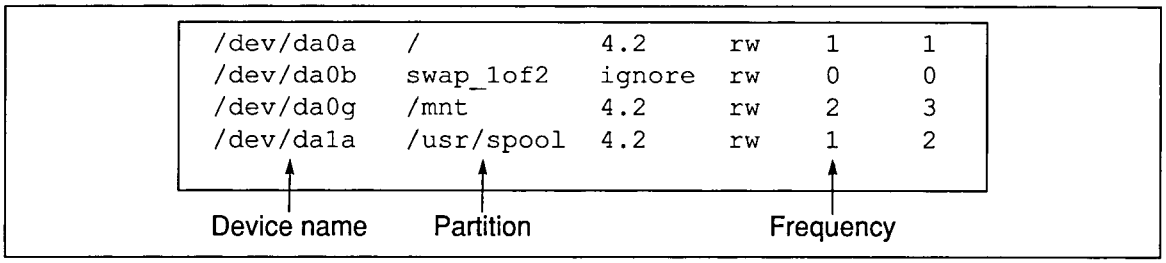
When planning a back-up schedule, weigh the amount of time required to perform a backup against the cost of losing data on the file systems, if their data is lost. Perform the following steps to develop a schedule for backing up the file systems in your computer:

Step 1 List the file systems in your system. This information is available in the /etc/fstab file. Enter:

```
% less /etc/fstab
```

The fstab file describes all file systems, whether they are mounted or unmounted, as well as the swap partitions. Figure 67 illustrates an example of the contents of the fstab file.

Figure 67 Example fstab file



The following information available in this file is helpful in determining a backup schedule:

- Device name
- Partition
- Frequency

Frequency is a system manager recommendation on how often the file system should be backed up to tape when the file system was created. This can be one of the following:

0 Never back up the file system.

1 Back up the file system daily (this is the recommended frequency).

2 Back up the file system every two days.

and so forth.

Caution

The frequency field is only used as a tool to know how often to dump the file system. It does not cause the file system to be dumped automatically.

Step 2 Judge what percentage of files change during a back-up period (usually a week) for each file system.

Step 3 Determine how often you want to dump each file system.

CONVEX recommends either an incremental or full backup of all file systems each working day, depending on the percentage of data that changes in a file system each week.

For example, in a file system with a moderate percentage of change each week, say 20%, you might perform a full backup each week and an incremental backup each working day. In a file system with small, infrequent changes, such as the root file system, you might perform incremental backups every day and a full backup only once every other week.

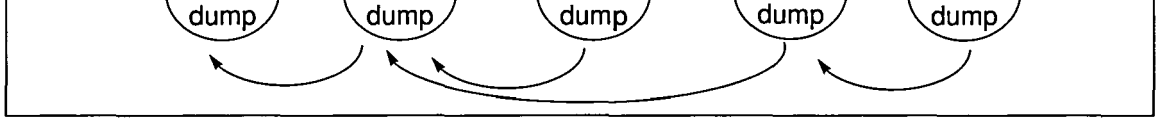
When planning the frequency of full backups, measure the time required to perform incremental backups against the time required to perform a full backup. Also, measure the time required to recover a file system from multiple incremental dump tapes, if it becomes necessary to recover a file system, against the time required to perform a full backup.

Step 4 Determine at what dump level you will take the incremental dumps.

Each file system backup is assigned a dump level between 0 and 9. Level 0 specifies an entire file system dump. Levels 1 through 9 specify incremental dumps. Levels are not necessarily assigned in consecutive order, although they can be. That is, one day you may perform a level 2 dump, the next day a level 5 dump, the next day another level 5 dump, the next day a level 6 dump, and so on.

Dump levels determine which files to back up during an incremental dump. An incremental dump backs up all files changed since the previous dump of a lower level. For example, a level 3 dump backs up all files changed since the most recent level 0, 1, or 2 dump. That is, if the most recent dump was a level 0 dump, it backs up any files changed since the level 0 dump. But, if there is a level 0, 1, and 2 dump, it backs up any files changed since the last level 2 dump. Figure 68 illustrates this.

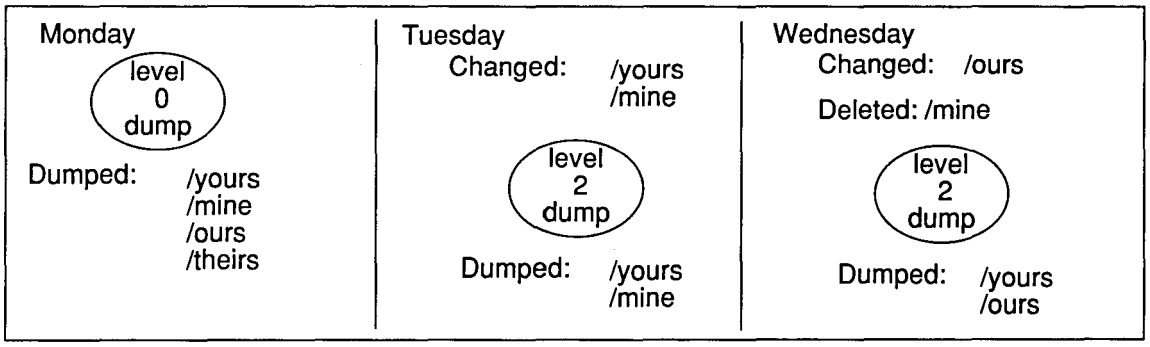
Figure 68 Incremental dumps using levels to determine which files to dump



In Figure 68, a level 2 dump is performed on two consecutive days. In this case, the second level 2 dump is a superset of the first level 2 dump because both incremental dumps back up all files changed since the previous level 1 dump.

The second level 2 dump may include more files, because any files that are changed after the first level 2 dump but before the second level 2 dump are included on the second level 2 dump, and not on the first level 2 dump. Furthermore, if some files changed before the first level 2 dump, but were deleted before the second level 2 dump, they will appear on the first level 2 dump but not the second level 2 dump. Figure 69 illustrates this.

Figure 69 Consecutive same-level incremental dumps

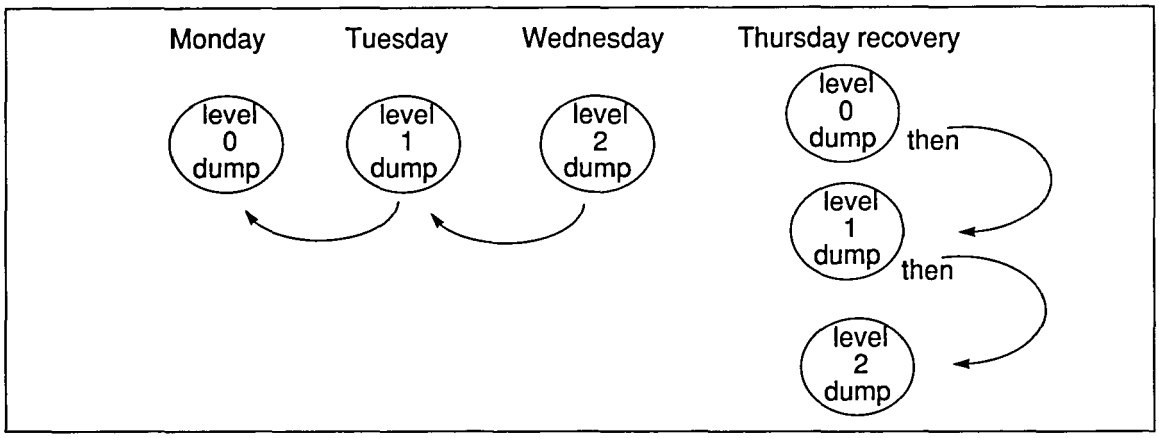


Before you choose dump levels for incremental backups, you should understand how files are restored from dump tapes. Consider the scenario where:

- Monday, a level 0 full dump is taken.
- Tuesday, a level 1 incremental dump is taken.
- Wednesday, a level 2 dump is taken.
- Thursday, the file system becomes corrupted and must be restored.

From this scenario, it would be necessary to restore the dump tapes from Monday, Tuesday, and Wednesday in order to recover the system on Thursday. See Figure 70 for an illustration of this.

Figure 70 Recovering from tape, scenario 1

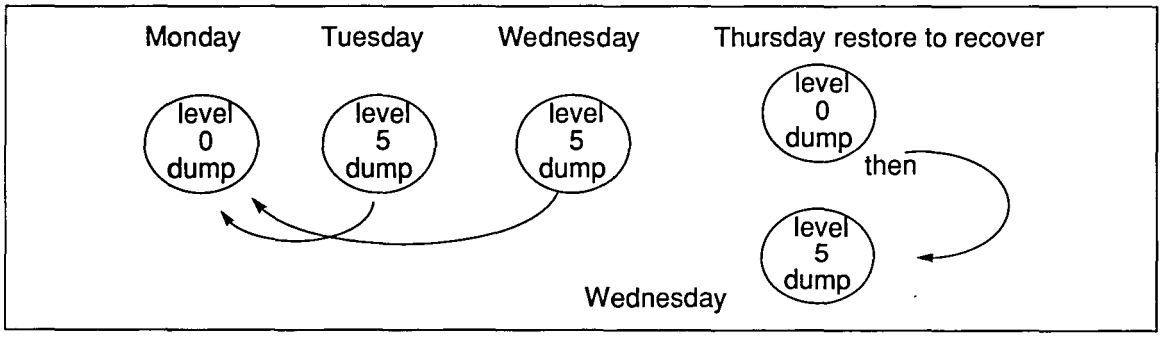


On the other hand, consider the scenario where:

- Monday, a level 0 full dump is taken.
- Tuesday, a level 5 incremental dump is taken.
- Wednesday, a level 5 dump is taken.
- Thursday, the file system becomes corrupted and must be restored.

From this scenario, it would be necessary to restore the dump tapes from Monday and Wednesday in order to recover the system on Thursday, because each level 5 dump backs up files changed since the level 0 full dump. See Figure 71 for an illustration of this.

Figure 71 Recovering from tape, scenario 2



CONVEX recommends that you use the same level for incremental dumps each time. The time required to perform the same level dump each day is slightly longer than if they were different level dumps, but the time required to restore the files is significantly less.

Step 5 Schedule which day of the week you will take full and incremental dumps for each file system.

If you are scheduling backups for multiple file systems, you can equally distribute the time spent taking dumps over the backup cycle by scheduling full dumps of file systems during different days of the week and incremental dumps on all file systems daily. For example, assuming the following file systems, /dev/rda0a, /dev/rst3, /dev/rst2, /dev/rst1, /dev/rda2d, /dev/rda2e, use the following schedule to equally distribute the time spent taking back ups throughout the weekly back-up cycle:

- Monday perform full dumps of /dev/rda0a and /dev/rst3 and incremental dumps of all file systems.
- Tuesday perform a full dump of /dev/rst2 and incremental dumps of all file systems.
- Wednesday perform a full dump of /dev/rst1 and incremental dumps of all file systems.
- Thursday perform a full dump of /dev/rda2d and incremental dumps of all file systems.
- Friday perform a full dump of /dev/rda2e and incremental dumps of all file systems.

This schedule runs incremental dumps on all file systems each day, including incremental dumps on those file systems completely dumped on the same day.

Step 6 Create back-up scripts for each day of the week. You can automate your back-up schedule by creating shell scripts. For example, assuming the schedule shown in Step 5, you can create the shell scripts shown in Figure 72 for automating back-ups.

In Figure 72 is a shell script for each day of the week and a shell script to perform incremental back-ups. See the dump(8) man page for details on the `dump` command.

Step 7 Determine an archive schedule. back-up tapes should be saved for a period of time and be available for more than one week past. For example, archive back-up tapes according to the following schedule:

- Save daily back-up tapes for three weeks. You can reuse tapes from week 1 for week 4 backups.
- At the end of each month, make a month-end backup and place it in the archive for an entire quarter.
- At the end of each quarter, make a quarter-end backup and place it in the archive for an entire year.

- At the end of each year, make a year-end back-up and place it in a permanent archive.

Figure 72 Sample back-up scripts

Monday

```
# full dumps on /dev/rda0a, /dev/rst3
dump 0nGfu /dev/rmt16 /dev/rda0a
dump 0nGfu /dev/rmt16 /dev/rst3
daily.incr
```

Tuesday

```
# full dump on /dev/rst2
dump 0nGfu /dev/rmt16 /dev/rst2
daily.incr
```

Wednesday

```
# full dump on /dev/rst1
dump 0nGfu /dev/rmt16 /dev/rst1
daily.incr
```

Thursday

```
# full dump on /dev/rda2d
dump 0nGfu /dev/rmt16 /dev/rst2d
daily.incr
```

Friday

```
# full dump on /dev/rda2e
dump 0nGfu /dev/rmt16 /dev/rda2e
daily.incr
```

daily.incr

```
# Incremental dump of all file systems
# tape 1
dump 5Gfu /dev/rmt20 /dev/rda0a
dump 5Gfu /dev/rmt20 /dev/rst3
dump 5nGfu /dev/rmt16 /dev/rst2
# tape 2
dump 5Gfu /dev/rmt20 /dev/rst1
dump 5Gfu /dev/rmt20 /dev/rda2d
dump 5nGfu /dev/rmt16 /dev/rda2e
```

Setting up the line printer system

5

The line printer system is a collection of programs and files for managing printer operations. The system can handle multiple printers, including laser printers and raster output devices such as Versatec plotters, multiple spooling queues, local and remote printers, and printers attached through serial lines.

This chapter discusses how to set up the line printer system. For information on managing the line printer system, refer to the chapter "Managing the line printer system," in the *Managing ConvexOS: Operations Guide*. For information on installing new printers, refer to Chapter 2, "Adding devices," in this guide. Error messages are listed in the appendix "Line printer system error messages," in the *Managing ConvexOS: Operations Guide*.

Printcap file

The /etc/printcap file is a master database containing descriptions for printers that can be accessed from your machine. A sample printcap file is shown in Figure 73.

Figure 73 Sample /etc/printcap file

```
lp|printer|local serial printer:\
    :lp=/dev/tty0a:br#9600:\
    :sd=/usr/spool/lpd:\
    :of=/usr/lib/lpf:if=/usr/adm/lpd-errs:
lp1|parallel printer:\
    :lp=/dev/lp1:sd=/usr/spool/lp1:\
    :af=/usr/adm/lpd-acct:if=/usr/lib/lpf:
rm1|remote printer on host "c2":\
    :lp=:rm=c2:rp=lp:sd=/usr/spool/rm1:
imaspp|generic Imagen serial printer:\
    :lp=/dev/tty1a:br#9600:\
    :cf=/usr/local/lib/icif.s:\
    :df=/usr/local/lib/idvi.s:\
    :tc=imaprint:
```

Each entry in this file represents one printer. Entry format is

printer_name|[*printer_name*|...] : :*field=string*[:*field#number...*] [:*field*]

where

printer_name The name or list of names used to refer to the printer. Printer names are separated by the pipe (|) character. You can refer to the printer by any of these names.

Fields specify characteristics for the printer, always preceded by a colon (:). These can be one or more of the characteristics listed in Table 15.

:*field=string* Fields of type string use this format.

:*field#number* Fields of type number use this format.

:*field* Fields of type boolean use this format. They are true if present, false if not.

Also note the following about the /etc/printcap file format:

- Entries that span multiple lines must have a colon followed by a backslash character at the end of each line.
- Be sure there are no additional spaces after the backslash. This can be verified by issuing the `:set list` command in the `vi` editor or by listing the printcap file using the `cat -e` command.

- Continued lines begin with a white space or tab character.
- The last line in each entry is terminated with a colon.

Table 15 Fields in the /etc/printcap file

Name	Type	Description
af	string	Path name of file where printer use data is stored for accounting system. This can be the same for all printers or different for each printer.
br	number	Baud rate—for printers attached to a tty line.
cf	string	Path name of `cifplot` data filter.*
df	string	Path name of `tex` data filter (DVI format).*
fc	number	Clears certain flags set for tty line, which, if set, might cause problems printing to the printer. See the printcap(5) and tty (4) man pages for more details.
ff	string	String to send for form feed.
fo	boolean	Prints a form feed when device is opened.
fs	number	Sets flags for tty line,which if clear, might cause problems printing to the printer. See the printcap(5) and tty(4) man pages for more details.
gf	string	Path name of graph data filter (plot (3X) format).*
ic	boolean	Driver supports (nonstandard) ioctl to indent printout.
if	string	Path name of filter that does accounting.*
lf	string	Path name of file where printer errors are logged instead of the console. This can be the same for all printers or different for each printer. This is true only for those errors that write to standard error.
lo	string	Path name of lock file. The default name is locked in the spool directory. This file is used to lock a printer for synchronized queueing. This prevents two files being queued to the spool directory at exactly the same time, one overwriting the other.
lp	string	Path name of device to open for output. For a printer connected directly to the local machine, this would be the name of the special device file represented by the printer, for example, /dev/lp1. For a printer connected through a terminal server connection (tty line), this would be the special device file representing the tty line to which the printer is connected, for example, /dev/tty0a.
mx	number	Maximum file size (in BUF-SIZ blocks). If a larger file is submitted for printing, the print job is rejected.

Table 15 Fields in the /etc/printcap file (continued)

Name	Type	Description
nf	string	Path name of ditroff data filter (device independent troff).*
of	string	Path name of filter program that processes output.*
pl	number	Page length (in lines) of printed page.
pu	boolean	Propagate user information to filters, such as user ID, group ID, and username.
pw	number	Page width (in characters) of printed page.
px	number	Page width, horizontal (in pixels).
py	number	Page width, vertical (in pixels).
rf	string	Path name of the filter to use when printing FORTRAN-style text files.*
rm	string	Path name of remote machine where remote printer is physically attached.
rp	string	Path name of remote printer physically attached to the remote machine.
rs	boolean	Restricts remote users of a printer to those with local accounts. If a user without a local account submits a remote job, the job is rejected.
rw	boolean	Opens the printer device for reading and writing.
sb	boolean	Short banner prints (one line only)
sc	boolean	Suppresses a request for multiple copies with the lpr command line. If more than one copy is requested, only one copy is printed.
sd	string	Path name of the spool directory for the printer. There should be a separate spool directory for each printer.
sf	boolean	Suppresses printing of burst page header.
tc	string	Specifies the name of an entry in /etc/printcap that has common characteristics with this printer. The system uses the characteristics defined in the specified entry, unless the characteristic is specified in this entry. If you use the tc field, it must appear last in the field list.
tf	string	Path name of troff filter (cat phototypesetter).*
tr	string	Contains string to print when queue empties and an lpq is executed.
vf	string	Path name of raster image filters. *
xc	number	Clears local mode bits if lp is a tty. See the printcap(5) and tty(4) man pages for more details.

Table 15 Fields in the /etc/printcap file (continued)

Name	Type	Description
xs	number	Set local mode bits if lp is a tty. See the printcap(5) and tty(4) man pages for more details.
*See Table 16 for a list of specialized ConvexOS filters.		

Table 16 lists specialized filters that are available with ConvexOS and their purposes. These filters are found in /usr/lib.

Table 16 ConvexOS specialized filters

Filter	Description
flpf	FORTRAN filter for FORTRAN-style carriage control.
lpf	nroff filter.
necf	Changes newlines to carriage returns and paginates text.
vpf	Varian/Versatec filter.
vpsf	Versatec filter for wide listings.
vdmp	cifplot data filter for Varian/Versatec.
vpltdmp	vplot data filter for Varian/Versatec.
vplotf	Standard graphics filter for Varian/Versatec.

Output filters

Output filters receive text as standard input. After processing, they send text to standard output, which is the printer device. The line printer system uses output filters for two purposes: to perform accounting functions and to handle device dependencies of different printer types. Filters routinely:

- Initialize the printer
- Process special characters
- Send text to the printer
- Process page parameters such as length and width

The name of the standard output filter is specified in the `of` field in the printcap file. A standard filter is useful when all text must pass through some filter, regardless of where the text originated.

Some devices, as well as certain sources of text such as `troff` and TEX, require a higher degree of filtering than provided by the standard filter. For this reason, several specialized filters are available.

The filters supplied with the line printer system handle printing and accounting for most printer types, including the Benson-Varian, and the wide (36-inch) and narrow (11-inch) Versatec printer/plotters. For other devices or accounting methods, it may be necessary to create a new filter. An example of a printer that requires output filters is the Benson-Varian, shown in Figure 74.

Figure 74 Output filter entry in /etc/printcap

```
va|varian|Benson-Varian:\
    :lp=/dev/va0:sd=/usr/spool/vad:\
    :of=/usr/lib/vpf:tf=/usr/lib/rvcat:mx#2000:pl#58:tr=\f:
```

The fields related to filters included in this example are explained below:

`of` Specifies the name of the standard filter.

`tf` Specifies that the /usr/lib/rvcat filter is used to print troff output. This filter is required to set the device to print mode for printing text, and plot mode for printing `troff` files and raster images.

`pl` Sets the page length to 58 lines for 8.5-inch by 11-inch fan-fold paper.

Printer accounting can be enabled by specifying an accounting filter with the `if` field and the name of an accounting file with the `af` field. Standard filters are not intended to perform accounting. If both the standard filter and another filter are specified in the printcap entry, the standard filter only prints the banner page and stops; the other filters are then allowed access to the printer.

To enable accounting, the Varian entry would be augmented with an `if` filter as shown in Figure 75. The `af` field specifies the file to store the accounting data.

Figure 75 Enabling printer accounting with the `af` filter

```
va|varian|Benson-Varian:\
    :lp=/dev/va0:sd=/usr/spool/vad:\
    :of=/usr/lib/vpf:if=/usr/lib/vpf:tf=/usr/lib/rvcat:\
    :af=/usr/adm/vaacct:mx#2000:pl#58:tr=\f:
```

Setting up a new printer

ConvexOS comes with the necessary line printer programs installed and with the default line printer queue, /usr/spool/lpd, created. To configure a new printer and create a separate spooling directory for queuing jobs to the printer, complete the following steps.

Step 1 Log in as the superuser.

Step 2 Define the printer to the system by editing the /etc/printcap file on the machine where the printer is physically connected. The information you place in this file depends on whether you are adding a serial or parallel printer.

If you are adding a printer to a serial port, specify the proper communication parameters. Figure 76 shows an example entry for a printer connected to a 9600-baud serial port.

Figure 76 Sample /etc/printcap entry for serial printers

```
lp|printer|local serial printer:\
    :lp=/dev/tty0a:br#9600:\
    :sd=/usr/spool/lpd:\
```

Each field included in this example is explained below:

`lp` Specifies the file name to open for output. In this case, specify the special device file representing the tty line to which the printer is connected, for example, /dev/tty0a.

`br` Sets the baud rate for the tty line.

`sd` Specifies the spool directory. There should be a separate spool directory for each printer.

`of` Specifies the filter program to use for printing files. See the section titled, "Output filters," in this chapter for information on the type of filters available. If these do not suffice and you need to write your own filter, see the section titled, "Creating a filter," in this chapter.

`lf` Specifies where to write errors if you do not want them sent to the console. Most errors from `lpd` are logged using the `syslog` facility and will not be logged in the specified file. Only those errors that write to standard error are written in the file specified by `lf`. For more information on the `syslog` utility, refer to Chapter 13, "Setting up log files."

If you are adding a printer connected to a parallel port, baud-rate entries or terminal mode settings are not required. Figure 77 shows an entry for a printer on a parallel port.

Figure 77 Sample /etc/printcap entry for parallel printers

```
lp1|parallel printer:\
    :lp=/dev/lp1:sd=/usr/spool/lp1:\
    :af=/usr/adm/lpd-acct:if=/usr/lib/lpf:
```

Each field included in this example is explained below:

`lp` Specifies the file name to open for output. In this case, you should specify the special device file represented by the printer, for example, /dev/lp1.

`sd` Specifies the spool directory. There should be a separate spool directory for each printer.

`af` Specifies the path name of the file where printer use data is stored for the accounting system. See Chapter 8, "Setting up the accounting system," for details on setting up this file to receive data.

`if` Specifies the path name of the filter that does accounting. See Table 16 for a list of ConvexOS specialized filters.

Step 3 If you are adding a printer connected to a serial port, edit /etc/ttys. Mark the port you specified in Step 2 as `off`.

Step 4 Use the `mkdir` command to create the spool directory for the new printer. Enter:

```
# mkdir /usr/spool/spool_directory
```

where

spool_directory is the name specified in the `sd` field of the /etc/printcap file.

Step 5 Change the user and group ownership of the spool directory to lpr, and set access permissions so that owner and group have read, write, and execute permission on the directory. To do this, enter:

```
# chall -o lpr -g lpr -m 770 /usr/spool/spool_directory
```

Step 6 Enable the queue with the `lpc` command. Enter

```
# lpc enable printer_name
```

where

printer_name is the name specified in the `lp` field of the /etc/printcap file.

Step 7 Start the printer daemon. Enter:

```
# lpc restart printer_name
```

Step 8 If the printer will be accessed from remote machines, place the following lines in the /etc/printcap file on each machine that will access the printer remotely. Otherwise, skip this step.

```
rm1|remote printer on convexhost:
lp=:rm=convexhost:rp=lp:sd=/usr/spool/convexl
pd:
```

In this example, output will be sent to the printer named lp on the machine convexhost. Each field included in this example is explained below:

`lp` Leave this field empty. This indicates that the printer is a remote printer and that no local special device file needs to be opened.

`rm` Specifies the name of the remote machine where the printer is physically connected. In this case, the remote machine is convexhost. This name must appear in the /etc/hosts database as it is specified here. Refer to Step 9 for details.

`rp` Specifies the name of the printer physically connected to the remote machine.

`sd` Specifies the spool directory on the local machine where print jobs are queued for printing. In this case, /usr/spool/convexlpd is specified instead of the default directory /usr/spool/lpd.

Step 9 If you performed Step 8, add the name of the remote machine to the /etc/hosts file, if it is not already there. Figure 78 illustrates an example /etc/hosts file entry.

Figure 78 Example /etc/hosts file entry

```
130.168.71.160  sunny   # any comment
```

Each line in this file represents one entry; each entry represents one host. The format of the /etc/hosts file is

internet_address official_name [*aliases* ...] [*#comment*]

where

internet_address is the official Internet address for this host.

official_name is the official name for this host, as specified with the `hostname` program.

aliases is an unofficial name or list of unofficial names for this host.

#comment is a comment about this host.

Step 10 If you are setting up a printer that will be accessed by remote hosts, create or modify the /etc/hosts.equiv file on the machine where the printer physically resides.

Step 11 Each line in this file represents one remote host. Figure 79 illustrates an example /etc/hosts.equiv file. Each line in this file represents one entry and should be the fully qualified domain name (FQDN).

Figure 79 Example /etc/hosts.equiv file

```
opekoe.tea.com
mint.tea.com
rosehips.tea.com
```

Creating a filter

Use the following specifications to write your own filter.

- Filters are spawned by `lpd`; their standard input is the data to be printed, and their standard output is the printer. Standard error is set to the file specified by the `lf` field in the /etc/printcap file.
- A filter must exit with a value of `0` (zero) if there were no errors, `1` if the job should be reprinted, and `2` if the job should be thrown away. When `lprm` wants to remove a job that is currently printing, it sends a `SIGINT` signal to all filters and their descendents. This signal can be caught by filters that need to perform cleanup operations such as deleting temporary files or resetting the printer.
- Arguments passed to a filter depend on its type:
  - The `of` filter is called with the following arguments:

    filter -w*width* -l*length*

    The *width* and *length* values come from the `pw` and `pl` fields in the printcap database.
  - The `if` filter is called with the following arguments:

    filter `[-c]` -w*width* -l*length* -i*indent* -n*login* -h*host accounting_file*

    The `-c` flag is optional and is supplied only when control characters are to be passed uninterpreted to the printer (when the `-l` option of `lpr` is used to print the file). The `-w` and `-l` parameters are the same as for the `of` filter. The `-n` and `-h` parameters specify the login name and host name of the job owner. The last argument is the name of the accounting file from /etc/printcap.
  - All other filters are called with the following arguments:

    filter -x*width* -y*length* `-n` *login* `-h` *host accounting_file*

    The `-x` and `-y` options specify the horizontal and vertical page size in pixels (from the `px` and `py` entries in the printcap file). The rest of the arguments are the same as for the `if` filter.

Controlling access

The line printer system maintains protected spooling areas so that users cannot circumvent printer accounting or remove files. The strategy used to maintain protected spooling areas is:

- The `lpr` program runs as suid lpr and sgid lpr. The `lpr` utility verifies, through an `access` system call, whether the user running `lpr` can read files. The sgid lpr sets up proper ownership of files in the spooling area for `lprm`.
- Control files in a spooling area are created with ownership and group of lpr; their mode is set to 0660. This ensures that control lines are not modified by a user and that no user can remove files except through `lprm`. Refer to the chmod(1) man page for information on file access modes.
- The programs `lpq` and `lprm` run as sgid to lpr. The programs `lpc` and `lpmv` run as suid lpr and sgid lpr to access spool files.
- Only user and group lpr can write to the spooling area.
- The printer daemon `lpd` runs as root to write to the printer devices and to bind Internet sockets to reserved ports.
- The printer server, `lpd`, uses the same verification procedures as `rshd` in authenticating remote clients. A remote host that wants to use a printer on the local host must be present in the/etc/hosts.equiv file.

Setting up a UUCP connection

6

UUCP (UNIX-to-UNIX Communications Protocol) is an intermachine communication system that can be run over direct serial lines, network connections, or ordinary telephone lines. It supports two operations: file copying and remote command execution.

Primarily, UUCP acts as a transport mechanism for electronic mail and news. Normally, users do not take advantage of UUCP's facilities directly; most UUCP services are transparent to a user.

UUCP software executes requests in a batch-oriented, store-and-forward manner. Requests for file transfer or remote command execution are not executed immediately, but are spooled for execution when communication is established between the two systems. Depending on your setup, you can establish communication immediately or wait until a later time when telephone rates are lower.

Each system on a UUCP network has a set of files that describe other systems connected to it. Creating these files is the major task of the system manager; once communication links are established, UUCP requires minimal supervision and administration.

This chapter describes the tasks required to create UUCP system files and configure your system to communicate on the UUCP network. These tasks are:

- Configuring modem connections
- Creating files necessary to UUCP
- Controlling remote access

See the appropriate section in this chapter for details on how to perform each task.

Caution

If you are setting up UUCP over Ethernet, first read the uucico(8) man page.

Configuring modem connections

It is necessary to describe to the system what modem connections are available for use by `uucp`. Perform the following steps to describe your modem connections:

Step 1 Set up and configure your modems.

How you configure your modems depends on whether you want an active system (dial-out only), passive system (dial-in only), or both active and passive. Refer to Chapter 2, "Adding devices," for information about installing modems and creating dial-in passwords.

Step 2 Log in as the superuser.

Step 3 Describe your modems in the/usr/lib/uucp/L-devices file. This file determines which devices are available to `uucico`, the UUCP protocol daemon. UUCP commands search through L-devices until an entry is found that matches its requirements. If that device is unavailable, it uses the next matching entry. An example L-devices file is shown in Figure 80.

Figure 80 Example L-devices file

```
ACU     cua0    cua0    300     vadic
ACU     cua0    cua0    1200    vadic
ACU     cua1    cua1    300     vadic
ACU     cua1    cua1    1200    vadic
DIR     tty0a   tty0a   9600    direct
```

caller *device* *call_unit* *class* *dialer*

The format of this file is:

caller device call_unit class dialer `[`*expect/send*`]` ...

where

caller indicates the type of connection. This can be one of the following:

`ACU`	Automatic Call Unit or autodialing modem[1]
`DIR`	Direct connection
`PAD`	X.25 PAD connection
`PCP`	GTE Telenet PC Pursuit
`TCP`	Berkeley TCP/IP connection

[1]Automatic Call Units were used in the past when autodialing modems were not yet available. Although some sites still use ACUs, today most sites are using autodialing modems.

device specifies the device file for data. This can be tty*??* or cua*n*. The name should be as it appears in the /dev directory.

call_unit is an optional second device file name. True automatic call units use a separate device file for data and for dialing; the *device* field specifies the data port, while *call_unit* specifies the dialing port. If *call_unit* is unused, it must not be left empty. Insert a dummy entry as a placeholder, such as `0` or `unused`.

class is the baud rate of the device.

dialer indicates the brand of modem or a direct connection. Valid entries are:

`hayes`	Hayes Smartmodem 1200 and compatible modems.
`hayes2400`	Hayes Smartmodem 2400 and compatible modems.
`maxwell`	Maxwell 1200VP modem.
`trailblazer`	Telebit Trailblazer 9600 baud modem. The *class* for this modem must to set to 0.
`va212`	Racal-Vadic 212 modem.
`vadic`	Racal-Vadic 3450 and 3451 Series modems.
`direct`	Direct connection. Use this for direct serial links between systems.

expect/send is an optional chat script for sending commands to a smart port selector, or modem.

Refer to the L-devices(5) man page for more information.

Creating files necessary to UUCP

Before you can use the uucp facility, you must create the work files, data files, and executable files necessary for spooled transfers. These files are kept in the /usr/spool/uucp directory.

Perform the following steps to create the files necessary to uucp operations with the proper ownership and permissions:

Step 1 Log in as the superuser.

Step 2 Run the /usr/lib/uucp/UUCP_SETUP script by entering:

```
# /usr/lib/uucp/UUCP_SETUP
```

This script creates the following files and directories in /usr/spool/uucp:

AUDIT	Directory for audit trail files (one per site)
C.	Directory for command (C.) files
D.	Directory for data (D.) files
D.*hostname*	Directory for local D. files (where *hostname* is the name of your local machine)
D.*hostname*X	Directory for local X. files (where *hostname* is the name of your local machine)
ERRLOG	Log file for assertion errors such as deadlock, permissions, and so on.
LCK	Directory for lock files
LOG	Directory for logging conversations
LOGFILE	Log file of UUCP activity
STST	Directory for status files
SYSLOG	Log file of UUCP file transfers
TM.	Directory for temporary (TM.) data files
X.	Directory for command execution (X.) files
XTMP.	Directory for temporary command execution files

Step 3 Make sure the ownership and access permissions of the /usr/spool/uucppublic directory are set properly. This directory is a temporary storage place for files being transferred to and from your machine and is commonly referred to as the public access directory.

To view ownership and access permissions on this directory, enter:

```
# ll -dg /usr/spool/uucppublic
```

Figure 81 illustrates the proper output for this command. This file should be owned by uucp and belong to group uucp. It should provide read, write, and execute access for owner, group, and other.

Figure 81 Access permissions for /usr/spool/uucppublic

```
drwxrwxrwx 2   uucp   uucp    512  Mar 19 10:28  uucppublic
```

Step 4 If the owner is uucp, skip this step. If it is not, enter:

```
# chown uucp /usr/spool/uucppublic
```

Step 5 If the file belongs to group uucp, skip this step. If it does not, enter:

```
# chgrp uucp /usr/spool/uucppublic
```

Step 6 If the access permissions allow read, write, and execute access for owner, group, and other, skip this step. If they do not, enter:

```
# chmod 777 /usr/spool/uucppublic
```

Step 7 View ownership and access permissions on the files in /usr/bin pertinent to uucp. To do this, enter:

```
# ll -g /usr/bin/uu*
```

Figure 82 illustrates output for this command.

Figure 82 Access permissions in /usr/bin

```
---s--s--x 1   uucp   uucp   147715     Dec 1 1989   /usr/bin/uucp
-rwxr-xr-x 1   root   bin    110528     Dec 1 1989   /usr/bin/uudecode
-rwxr-xr-x 1   root   bin    48220      Dec 1 1989   /usr/bin/uuencode
---s--s--x 1   uucp   uucp   62229      Dec 1 1989   /usr/bin/uulog
---s--s--x 1   uucp   uucp   54220      Dec 1 1989   /usr/bin/uuname
---s--s--x 1   uucp   uucp   81071      Dec 1 1989   /usr/bin/uupoll
---s--s--x 1   uucp   uucp   71667      Dec 1 1989   /usr/bin/uuq
---s--s--x 1   uucp   uucp   111455     Dec 1 1989   /usr/bin/uusend
---s--s--x 1   uucp   uucp   57218      Dec 1 1989   /usr/bin/uusnap
---s--s--x 1   uucp   uucp   147501     Dec 1 1989   /usr/bin/uux
```

Step 8 Make sure the ownership and access permissions on all UUCP programs in the /usr/bin directory are set properly:

- Ownership for `uuencode` and `uudecode` is set to user root and group bin.
- Access permissions on `uuencode` and `uudecode` are read, write, and execute for owner; and read and write for group and other; (mode 755).
- Ownership for all other UUCP programs is set to user uucp and group uucp.
- Access permissions for all other UUCP programs are set to execute for other (mode 001).
- All programs except `uuencode` and `uudecode` are set to setuid uucp and setgid uucp.

Step 9 View ownership and access permissions on the files in /usr/lib/uucp. To do this, enter:

```
# ll -g /usr/lib/uucp
```

Figure 83 illustrates output for this command.

Figure 83 Access permissions in /usr/lib/uucp

```
-rw------- 1   uucp   uucp    290       Jun 14 1989 L-devices
-rw------- 1   uucp   uucp    0         Apr 4 1984  L-dialcodes
-rw-rw-rw- 1   root   bin     137       May 1 18:06 L.aliases
-rw------- 1   uucp   uucp    76        Apr 17 1987 L.cmds
-rw-r--r-- 1   uucp   uucp    422       Oct 23 1989 L.sys
-rw-r----- 1   uucp   uucp    4         Apr 1 14:28 SEQF
-rw------- 1   uucp   uucp    35        Jun 3 1989  USERFILE
-rwx------ 1   uucp   uucp    868       Dec 1 1989  UUCP_SETUP
---s--s--x 1   uucp   uucp    300063    Dec 1 1989  uucico
---s--s--x 1   uucp   uucp    126426    Dec 1 1989  uuclean
---s--s--x 1   uucp   uucp    165253    Dec 1 1989  uuxqt
```

Step 10 Make sure the ownership and access permissions on these files are set properly:

- Ownership for all files is set to user uucp and group uucp.
- Access permissions on the L.sys file are read, write on owner, and read on group and other (644).
- Access permissions on the SEQF file are read, write on owner, and read on group.
- Access permissions on UUCP_SETUP are read, write, execute on owner (mode 700).

- Access permissions on `uucico`, `uuclean`, and `uuxqt` are set to execute for others (mode 001).
- `uucico`, `uuclean`, and `uuxqt` are setuid uucp and setgid uucp.
- Access permissions on all other files are read and write for the owner only (mode 600).

Note

The /usr/spool/uucp directory and its files are automatically set to the proper ownership and access permissions by the /usr/lib/uucp/UUCP_SETUP script. Nothing further is required for the files in this directory.

Controlling remote access

The UUCP system can be configured to operate as an active system, a passive system, or both. An active system is capable of dialing out to other systems on the UUCP network. A passive system does not dial out to other systems. Most UUCP setups operate as both active and passive by configuring dial-in and dial-out modems.

Perform the following steps to configure remote access to and from your system:

Step 1 Check to be sure a user account named *uucp* exists on your system. Issue the following command:

```
# grep uucp /etc/passwd
```

Note

This account is used by the local system to handle file transfers or remote command execution requests. Typically, the system is shipped with this account already created.

Step 2 Log in as the superuser.

Step 3 If this account does not exist, create it using the nu utility. Set the account up with the following information:

User Name:	uucp
User ID:	14 (this is the reserved user ID for the uucp user account)
Group Name:	uucp
Group ID:	40 (this is the reserved group ID for the uucp group)
Home Dir:	/usr/lib/uucp
Shell:	/bin/csh
Initial Password:	* for impossible password. No one will actually log in to this account, so there is no need to set a valid password.

Step 4 If you are setting up a passive system, or you want your system to operate as both active and passive, create user accounts using the nu utility for each remote system (client) that will be calling you. Again, refer to Chapter 7, "Setting up user accounts," on page 161, for details on how to set up user accounts. Provide the following information for each account:

User Name:	Unique account name. If possible, establish a naming convention such as U*sitename*, where *sitename* is the name of the remote site.

User ID:	Give each account a unique UID.
Group Name:	Set the group name for each account to uucp (GID 40).
Home Dir:	Set the path of the home directory to /usr/spool/uucppublic.
Shell:	Set the login shell to /usr/lib/uucp/uucico.
Initial Password:	Give each account a unique password.

Note

Be sure that remote sites are aware of the account name and password you assign, as this is the login name and password they must use to log into the system.

Step 5 Edit the /usr/lib/uucp/L.sys file to describe systems you communicate with.

Caution

Because this file contains the phone numbers and login passwords of remote systems, it should not be readable by anyone except uucp.

What you put in this file depends on whether you are setting up an active system, a passive system, or a system that is both active and passive.

If you are setting up a passive system, the L.sys file contains the host names of remote systems that will be calling your site. Figure 84 illustrates entries in the L.sys file in a passive system.

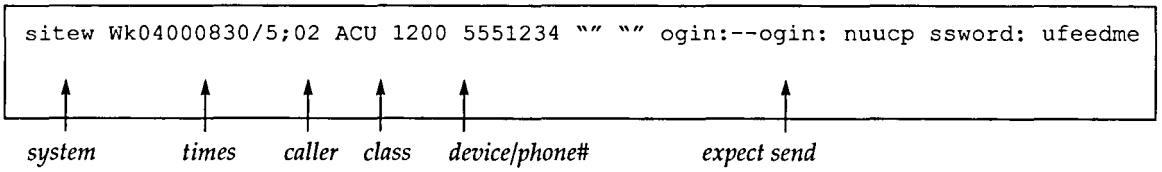
Figure 84 Example L.sys file for purely passive systems

```
sitew
sitex
sitey
sitez
```

For example, if a system called convex will be calling your site, enter the name convex in this file.

If you are setting up an active system, the L.sys file contains names of systems you will call and instructions on how to call them. The `uucico` daemon reads this file to determine how to call a remote system. Each line in the file describes how and when to access a particular host. Figure 85 illustrates an entry in the L.sys file in an active system.

Figure 85 Example L.sys file for active systems



The format of this file is

system times caller class device/phone# [expect/send] ...

where

system is the host name of the remote system.

times is a comma separated list of times when you can call the remote system. Commonly used to restrict long-distance calling to times when telephone rates are lower. Listed times are constructed as follows:

keyword [hhmm-hhmm]/grade;[retry_time]

The *keyword* entry is required. Valid entries are:

`Any`	Any day, any time
`Wk`	Weekdays
`Mo`	Monday
`Tu`	Tuesday
•	•
•	•
•	•
`Su`	Sunday
`Evening`	5 p.m. to 8 a.m. M-F, all of Sat and Sun
`NonPeak`	6 p.m. to 7 a.m. M-F, all of Sat and Sun
`Night`	11 p.m. to 8 a.m. M-F, all Sat, Sun to 5 p.m.

The optional *hhmm-hhmm* entry refers to hours and minutes and provides a time range that modifies *keyword*. A 24-hour clock is used; valid times are from 0000 to 2359. Do not enter a time range if you use the `Evening`, `NonPeak`, or `Night` keywords.

The */grade* entry is optional. This entry is composed of a slash followed by a single character (0 to 9, A to Z, or a to z) that specifies the grade (priority) of a request that can be transferred at this time. 0 is the highest grade; z is the lowest. Use lower grades for high volume jobs such as news. The grade of a particular request is assigned by the `uucp` or `uux` command.

Default grades are listed below:

`uux`	A
`uucp`	n
`mail`	C
`news`	d

retry_time specifies when a failed connection can be retried. The retry entry is a two-digit number that specifies how many minutes to wait before attempting the call again. The number must be preceeded by a semicolon (;). This entry is optional, though recommended.

caller is the type of device to use for the call. If the remote system cannot be called, enter `Slave` as the device type. Valid entries are:

`ACU`	Automatic call units or autodialing modems
`DIR`	Direct connections
`PAD`	X.25 PAD connection
`PCP`	GTE Telenet PC Pursuit
`TCP`	TCP/IP connection

class is the baud rate to use. For terminals or modems, it is the port number for TCP/IP.

device/phone# is the phone number of the remote system or the device name for direct connections. Phone entries can contain abbreviations that are defined in the L-dialcodes file.

expect/send is a string describing the initial conversation between two machines. It describes the login password and special character sequences needed to complete the login procedure.

The format of the *expect/send* pair is:

[*expect-timeout-send* [*-expect-timeout-send...*]]

The value specified for *expect* is compared against incoming text from the remote host. This can be:

string — Any string of text.

Double quotes ("") — Interpreted as expect nothing. The *send* string is transmitted regardless of what is received.

`ABORT` *string* — Abort on receiving the specified *string*. Once set, if that string is received any time prior to the completion of the entire expect/send script, `uucico` aborts as if the script timed out. This is useful for trapping error messages such as "Host Unavailable" or "System is Down" from port selectors or front-end processors.

Escape sequence — Escape sequences that can be used in *expect* or *send* strings are listed in Table 17.

Table 17 L.sys escape sequences for expect/send pairs

Escape	Description
`/b`	Generate a three-tenths of a second BREAK.
`/b`*n*	Generate *n*-tenths of a second BREAK, where *n* is any single-digit number.
`/c`	Suppress the `/r` at the end of a *send* string.
`/d`	Delay; pause for 1 second (send only).
`/r`	Carriage return.
`/s`	Space.
`/n`	Newline.
/xxx	Where *xxx* is an octal constant that represents the corresponding ASCII character.

If *expect* is not matched within a period specified by *timeout*, the system assumes the match failed. The *timeout* period is optional and can be specified by appending the parameter *~nn* (where *nn* is the timeout time in seconds) to the *expect* string. The default period is 45 seconds.

send is sent back when *expect* is matched. By default, *send* is followed by a `\r` (carriage return). Possible keywords for the *send* string are listed in Table 18.

Table 18 L.sys keywords for *send* strings

Keyword	Description
`""`	Send a carriage return (same as CR).
`BREAK`	Generate a three-tenths of a second BREAK.
`BREAK`*n*	Generate *n*-tenths of a second BREAK, where *n* is any single-digit number.
`EOT`	Send an End-Of-Transmission character (ASCII /004). This usually causes a remote host to hang up.
`PAUSE`	Pause for 3 seconds.
`PAUSE`*n*	Pause for *n* seconds.
`CR`	Send a carriage return (same as "").
`NL`	Send a newline.
`P_ODD`	Use odd parity on future *send* strings.
`P_ONE`	Use parity one on future *send* strings.
`P_EVEN`	Use even parity on future *send* strings (default).
`P_ZERO`	Use parity zero on future *send* strings.

The *expect-send-expect* notation provides a limited loop mechanism; if the first *expect* string times out and fails, the *send* string between the hyphens is transmitted and `uucico` waits for the second *expect* string. This can be repeated indefinitely. When the last *expect* string fails, `uucico` hangs up and logs the failed connection.

For example,

```
"" ogin:--ogin: nuucp ssword: ufeedme
```

is executed as:

1. When the remote system answers, expect nothing ("").

2. Send a carriage return ("").
3. Expect the remote to transmit the string 'ogin:'.
4. If it does not transmit within 45 seconds, send another carriage return.
5. If it sends 'ogin:,' send it the string 'nuucp'.
6. Then expect the string 'ssword:'.
7. When 'ssword:' is received, send the string 'ufeedme'.

Notice that the first character of each string has been left off. This is because the first letter may be uppercase or lowercase, and because noisy telephone lines tend to affect the first character of a line more often than characters in the middle of the line.

Refer to the L.sys(5) man page for more information.

If your system is both active and passive, identify those sites you do not call by specifying `Slave` in the *type* field. An example is shown in Figure 86.

Figure 86 Example L.sys file for active and passive operation

```
sitew    Any     ACU    1200  5551234 "" \r ogin: Cxuucp word: secret
sitex    Never   Slave  300   null
sitey    Never   Slave  300   null
sitez    Never   Slave  1200  null
```

Step 6 If you are setting up an active system, edit the /usr/lib/uucp/L-dialcodes file to map dialing prefixes to a specific dialing sequence in the L.sys file. The L-dialcodes file is useful when you want to talk to a number of systems at one site that have a common dialing sequence. By using this file to describe the dialing prefix, you can use simple dial-code abbreviations in the *phone#* field of the L.sys file. For example, a dialing prefix such as

```
anytown 1-515-789
```

in the L-dialcodes file might have a corresponding entry in L.sys with the first five fields as follows:

```
anywhere Any ACU 1200 anytown1234
```

When UUCP reads this entry, it refers to the L-dialcodes file and sends the dialing sequence:

```
1-515-7891234
```

Each dialing prefix in L-dialcodes should be listed on a separate line. The location is written in lowercase letters with no spaces between words (as in *anytown* rather than *any town*). Figure 87 shows an example L-dialcodes file.

Figure 87 Example L-dialcodes file

```
#phone number for anytown
#
anytown 1-515-789
#
```

Step 7 If you are setting up an active system, test the dial-out connections using `uucico` in debug mode by using the following format:

`uucico -x`*debug* `-r1 -s`*remotehost*

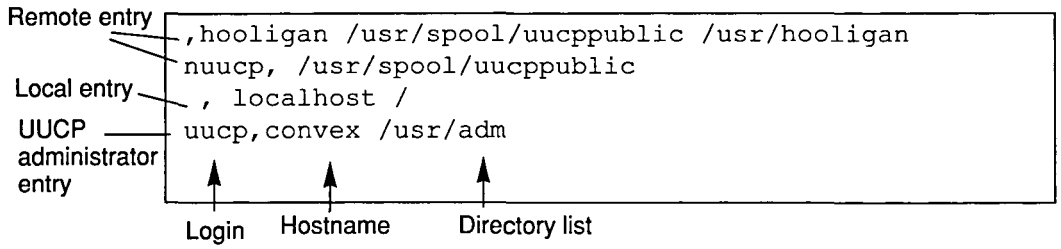
where *debug* is the debug level you want to use and *remotehost* is the name of the remote host. Debug level 5 is a good level for debugging connections. See the uucico(8) man page for details on debug levels and output.

Step 8 Edit the USERFILE to specify access to and from your system by local and remote users. The USERFILE controls:

- Which files can be accessed by a local user. A local file is subject to USERFILE restrictions and normal ConvexOS file permissions.
- Which files can be accessed by remote systems. This is important for machines configured for passive operation.
- The login name the remote system must use to talk to the local system.
- The callback option.

Figure 88 illustrates an example USERFILE.

Figure 88 Example remote entries in USERFILE



The format of USERFILE is

[*login*],[*hostname*] [c] *directory* [*directory...*].

where

login is the login ID of either a local user or the login name that will be used by a remote system. An entry must exist in the /etc/passwd file for this UUCP login name.

hostname is the host name of the remote system as it appears in the L.sys file. There must be a comma (but no space) separating the *login* and *hostname*, even if one or both fields are left blank.

c specifies that a callback should take place. The callback option causes the current conversation to end so the local machine can attempt to call the remote machine back. This provides security against intruders.

directory is a single directory path name or list of directory path names the user who is logging in with the specified *login* and from the specified *hostname* can access. This list should be as restrictive as possible.

Add these kinds of entries:

- Add an entry to the USERFILE for every remote machine to which you wish to allow access to your machine through uucp. For example:
  - In Figure 88, the second line represents a remote entry that allows anyone logging in as nuucp locally or from any remote host access to the /usr/spool/uucppublic directory, unless that host has an explicit USERFILE entry containing a system name (such as line1 in the example). For this to be true, this line must be the first occurrence in the file without a system name.
  - The first line in Figure 88 represents a remote entry that allows anyone logging in from host hooligan access to the /usr/spool/uucppublic and /usr/hooligan directories.
- Be sure there is at least one line in the USERFILE with a blank *login* field. This gives local users the ability to send outgoing file transfer requests to remote systems.
- Be sure there is at least one line in the USERFILE with the local *hostname* field. This gives local users the ability to request incoming file transfer requests from remote systems.

  These fields can occur in the same entry. For example, line 3 in Figure 88 allows all users (as specified by the blank *login*

field) from the local system to access all files under root (as specified by /). If you did not have a blank *login* entry like this, you would need to have a separate line for each user on your system.

Step 9 Edit the /usr/lib/uucp/L.cmds file to list commands that can be executed from a remote machine. Commands are listed separately on each line, and an optional PATH variable can be included on the first line of the file.

Caution

Carefully choose the commands you include. Giving remote sites access to commands can create security risks.

An example L.cmds file is shown in Figure 89.

Figure 89 Sample L.cmds file

```
PATH=/usr/local/bin:/bin:/usr/bin
rmail
nfrcv
nfxmit
rnews
ruusend
```

Step 10 Transfer of files or messages to your system sent via uucp occurs when you poll the sending host or it polls your system. This can be automated using crontab. If you are polling only one host, skip to Step 12 of this task. If you are setting up an active system and polling multiple hosts for messages sent via uucp, you can simplify polling by creating a script, like the one shown in Figure 90.

Figure 90 crontab script for polling remote sites

```
%!/bin/sh
% Poll these hosts periodically. Invoke this script
% from /.crontab (usually twice per day).

% Change the names of the hosts listed below for your configuration.

for i in host1 host2 host3
do
              /usr/bin/uupoll $i
done
```

Step 11 To fully automate polling, you can submit the script created in Step 10 to crontab so cron will make calls to remote systems at specified times. cron executes commands at specified dates and times according to the instructions found in the /.crontab file.

Figure 91 illustrates an example entry in the crontab file to execute the script created in Step 10 every Friday of each month, at 6:30 A.M.

Figure 91 Sample /.crontab file

```
0 06,18 * * * script_name
```

Each line of the crontab file represents one activity; each field in this line is 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	can be any number between 0 and 59.
hour	can be any number between 0 and 23.
date	can be any number between 1 and 31.
month	can be any number between 1 and 12.
day	can be any number between 1 and 7, where 1 equals Monday, 2 equals Tuesday, 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 one 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.

See the uucico(8C) man page for more information on polling.

Step 12 If you are polling only one host, you can add the following line to crontab. `cron` executes commands at specified dates and times according to the instructions found in the /.crontab file. For example:

```
30 06 * * * /usr/bin/uupoll hostname
```

In this example, the `uupoll` command is executed every day of each month, at 6:30 A.M. See Figure 91 for the format of the crontab file. For more information on using `cron`, refer to the cron(1) and crontab(5) man pages.

Setting up user accounts 7

Each person wishing to log into the CONVEX computer system and use its resources must have:

- An entry in the /etc/passwd file
- An entry in the /etc/group file

A user should also have a home directory containing at the very least a .login or .profile file that contains a series of commands initializing the user's environment and search path. However, this can also include other dot files such as .csh, .ksh, .exrc, and .logout.

Login routines use the /etc/passwd file to get information about the user, such as the user's home directory, password, and which shell to execute for the user. With this information, the system creates a shell process for the user and executes the dot files in the user's home directory.

The nu utility aides in creating the files and file entries required for a user to use the system. This chapter describes how to use the nu utility and provides more information on each of the required files.

Types of user accounts

A user account consists of all the information needed for a person (known as a user) to log into the computer system. There are two types of user accounts on the CONVEX system: ordinary user accounts and a superuser account.

There is only one superuser account on the system. The superuser, also known by the user name root, is a user who has privileged access to the computer system. The superuser is permitted to do any operation. The superuser has full read, write, and execute privileges for all files in the system, regardless of who owns them or their access privileges. Because superuser accounts bypass all system security measures, they are a security risk and must be handled with extreme caution.

Ordinary users have control over their own files only.

Each user account is identified by its user ID (UID). UIDs are the basis for:

- Accounting
- Controlling the use of disk space
- Accessing the file system
- Restricting access to privileged kernel operations, such as the request used to reboot a running system
- Controlling access to files and processes they own

A file is owned by only one user. To extend file access to multiple users, users are organized into groups. Groups allow two or more accounts to share access to files without granting file access privileges to the entire user community. This ability allows projects to be organized on a group basis. Refer to Chapter 1, "Security considerations," on page 1, for more details on protecting access to files.

Each user can belong to up to 16 groups. Each group is assigned a group identifier called GID. GIDs, like UIDs, are used for controlling access to files and resources.

The password file

User accounts are stored in the /etc/passwd file. Each user, in order to exist in the system, must have an entry in this file. The /etc/passwd file contains login information for each user in the system, including a user's encrypted password.

You can control selection of passwords by placing typing and/or password aging restrictions on the selection process. Password restrictions are used to increase security. They ensure that users participate in password security by selecting secure passwords (typing restrictions) and by changing those passwords regularly (password aging restrictions). If you specify typing restrictions for a user, any password selected must:

- Contain at least six characters
- Contain at least two alphabetic characters and one numeric or special character
- Not be the user's login name or any rotated permutation of it
- Differ from the previous password by a minimum of three characters

Table 19 lists the number of characters required in passwords, given the combination of characters used.

Table 19 Password length/character requirements

Types of characters used	Number required
Combination of upper-case, lower-case, and numeric	4
Combination of upper-case and lower-case	5
Monocase	6

If you specify password aging restrictions for a user, you can enforce the following rules:

- A password must remain unchanged for a minimum number of weeks. This means users cannot change back to their original password immediately after being forced to select a new one.
- A password remains valid for a maximum number of weeks. When the password is no longer valid, the user is prompted to set a new password.

- Temporary passwords, which are valid for one login, and other special passwords are possible using special age codes. This way, you can be assured that guest users are only users for the intended period of time.

An /etc/pwrestrict file must exist for password restrictions to apply. This file is created from the information stored in the /etc/passwd file. This is automatic when you use the `nu` utility to add new users. If you are adding users manually, you must run the `genrest` program to create this file.

Default user files

When you add a new account using the nu utility, all files stored in the /usr/skel directory are copied to the new user's home directory. The system is shipped with the following files in /usr/skel that control the user's working environment:

- .cshrc
- .login
- .logout
- .exrc

You can modify these files (or create additional user files such as .profile or .ksh) in this directory using an editor. Be sure these files contain the appropriate commands for your user environment before adding new users.

Note

Any changes you make to these files apply only to new users created after the changes. Existing user files are not changed.

This section describes each of these files, their purposes, and their default settings. See the *ConvexOS Primer* for information on the commands and variables you can place in these files.

Start-up default files

The shell that is initially started when a user logs in is specified in each user's record in the /etc/passwd file. (Shells are programs that read commands and execute them.) This start-up shell provides an initial working environment for a user. The default start-up program for ConvexOS is the C shell.

When a user logs in, the C shell start-up program reads and executes the commands in the .cshrc and .login files in the user's home directory. The start-up program first executes the commands in the .cshrc file and then those in the .login file. Directives in the .login file override those in the .cshrc file.

The .cshrc and .login files contain variables that control the user's environment. See the *ConvexOS Primer* for more information on shell variables.

.login file

The .login file is only read during a login routine (for example, `login`, `rlogin`, `xterm`). Therefore, you should place commands you want to execute only once during a login session in .login; .login generally includes terminal characteristic and environment variables. Figure 92 illustrates the /usr/skel/.login file shipped with ConvexOS.

Figure 92 .login file as shipped with ConvexOS

```
set mail = /usr/spool/mail/$user
stty crt erase "^H" kill "^U"
msgs -q
tset -Q
```

.cshrc file

Commands in the .cshrc file are read during a login routine and each time you start a C shell. Place commands you want to apply to each new shell in .cshrc. Figure 93 illustrates the /usr/skel/.cshrc file shipped with ConvexOS.

Figure 93 .cshrc file as shipped with ConvexOS

```
set path = (. ~/bin /usr/convex /usr/ucb /bin /usr/bin
set cdpath = (~)
set history = 20
set notify
umask 002
alias cd 'set old=$cwd; chdir \!*'
```

The .logout file

This file is not required by the system to execute a `logout` properly, however, it is useful in many environments. This file typically contains a `clear` command to clear the screen on a `logout`.

The .exrc file

This file is not required by the system to execute properly; however, it is used by the `vi` and `ex` utilities for customization purposes. This file typically contains key mappings and `vi`-specific variables.

Adding users

There are a number of ways you can add a new user to the /etc/passwd file: manually, interactively, or in batch. The following sections describe the procedures to perform for each method.

Adding users interactively using the nu utility

The nu utility automates adding new users. This utility automatically updates the following files when a user is added, eliminating the need to perform each step manually:

- /etc/passwd
- /etc/pwrestrict
- /etc/uidcount

The nu utility also creates a home directory for each user and copies the files in /usr/skel there. To add new users interactively using the nu utility, perform the following steps.

Step 1 Log in as the superuser.

Step 2 Check to be sure the files in /usr/skel have the desired variables.

Step 3 Using an editor, set up a file of default constants for the users you are adding. (This is an optional step.)

In interactive mode, the nu utility prompts for all information about a user. Some of this information is the same for each user you are adding. To eliminate the necessity of repeatedly typing the same value, you can create a file containing default values for these fields. The nu utility will use these values when adding a new user.

The default name for the file containing the constant values is /etc/nurc, although you are not restricted to this name. In fact, you can set up multiple files with different constants for different groups using a unique name for each file. Then, when you invoke the nu utility, you can specify an alternate file name using the -n option.

Each line in the default constants file represents a field with its default value. The format for this file is:

fieldname:value

fieldname can be one of the entries listed in Table 20. Entering only the field name in the file turns on the default value also listed in Table 20.

Table 20 Possible entries in default constants file

Field	Default	Description
uid		Identification number for user. This number takes the value from the /etc/uidcount file unless the `nouidfile` field is set; then it uses a number one greater than the highest UID already in use. Do not use numbers 0 through 99; these are reserved for use by CONVEX.
gid	EMPTY	Group identification number for user.
group	staff	Group name for user. Only use this field name if you do not use the gid field number.
directory	/mnt	Path under which home directory is placed. The home directory always has the same name as the login name.
protection	0755	File permissions placed on home directory.
shell	/bin/csh	Default login shell.
password	login name	User's initial password.
username	username	User's full name for the `finger` command.
office	office	User's office number for the `finger` command.
extension	extension	User's work telephone extension.
homephone	homephone	User's home phone number.
minwks	1	Minimum number of weeks for password aging.
maxwks	52	Maximum number of weeks for password aging.
diskquota	6, 8, 12, 15000	Soft limit for number of blocks a user can use, hard limit for number of blocks a user can use, soft limit for the number of inodes a user can own, hard limit for the number of inodes a user can own. (See Chapter 9, "Setting quotas on disk space use," for more details on hard and soft limits.)
skeleton	/usr/skel	Directory containing files to copy into home directory.
homedir	directory/login	Home directory to create for this user.
typed	OFF	Set password typing restrictions.

Table 20 Possible entries in default constants file (continued)

Field	Default	Description
aged	OFF	Set password aging restrictions.
quota	OFF	Initialize quotas for home directory file system.
nouidfile	OFF	Ignore the contents of the /etc/uidcount file. Use a number one greater than the highest UID already in use in the /etc/passwd file.
newsgroup	OFF	Is user in the Share scheduling group?
notshared	OFF	Is user in not-Shared scheduling group?

Step 4 Start the nu utility and enter the values for each new user, one user at a time. Enter

```
# nu [-n file_name]
```

where

file_name is the name of the file that contains default values. If you do not specify the -n option, the system uses the default file /etc/nurc.

The nu utility prompts for a login name of the user you are adding. The login name cannot be more than eight alphanumeric characters and by convention is lowercase. Do not use underscore, hyphen, or punctuation characters in the login name.

In interactive mode, nu prompts for all the information about a user. The default values are displayed for each field in square brackets. Press **RETURN** to accept the default value or enter the value you want to use instead. Figure 94 illustrates an example nu session.

Step 5 Customize the working environment (.cshrc and .login files) for each individual user if necessary. See the *ConvexOS Primer* for details on how to do this.

Figure 94 Example nu session

```
# nu
Login name: rocky
User id [800]:
Group id [staff]:
Home directory [/mnt/rocky]:
Home directory protection [0755]: 700
Login shell [/bin/csh]:

Changing the user information...
Default values are printed inside of [ ].
To accept the default, type <return>.
To have a blank entry, type the word 'none'.

Name [username]: Rocky Raccoon
Room number (Exs: 18A or 17B) [office]: 546
Office Phone (Ex: 223) [extension]: 798
Home Phone (Ex: 6610379) [homephone]: none

Rebuilding passwd database.

Password to be typed [N]:
Password subject to aging [N]: Y
          Enter the minimum period for the password [1]: 2
          Enter the maximum period for the password [52]: 4

Entering the user password...
New password:
Retype new password:

Rebuilding passwd and pwrestrict databases
```

When the user types the password, it is not echoed on the screen.

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

Adding users in batch using the nu utility

Using the nu utility, you can add several new users without interaction by creating a batch input file. Using nu in batch mode performs all the same actions as it does in interactive mode. Refer to the section, "Adding users interactively using the nu utility," on page 167, for the details on what nu does.

Invoked in batch mode, the nu utility uses two files for field definitions: the /etc/nurc or other user-named file containing default constant values, and a user-named batch file. Fields that are common to a large group of new users are placed in the /etc/nurc or user-named file containing defaults, and the user-specific fields are placed in the user-named batch file. The fields you can define for the batch file include all those available for the /etc/nurc file with one addition, a login field. See Table 20 for a list of possible fields. To add new users in batch mode, perform the following steps:

Step 1 Log in as the superuser.

Step 2 Check to be sure the files in /usr/skel have the desired variables.

Step 3 Using an editor, set up a file of default constants for the users you are adding. (This is an optional step.)

In interactive mode, the nu utility prompts for all information about a user. Some of this information is the same for each user you are adding. To eliminate the necessity of repeatedly typing the same value, you can create a file containing default values for these fields. The nu utility will use these values when adding a new user.

The default name for the file containing the constant values is /etc/nurc, although you are not restricted to this name. In fact, you can set up multiple files with different constants for different groups using a unique name for each file. Then, when you invoke the nu utility, you can specify an alternate file name using the -n option.

Each line in the default constants file represents a field with its default value. The format for this file is:

fieldname:value

fieldname can be one of the entries listed in Table 20. Entering only the field name in the file turns on the default value also listed in Table 20.

Step 4 Using an editor, set up a batch file. Call it whatever you like. Place user-specific fields in the this file. Each line in the file represents a user; each line includes information specific to that user; each field is separated by colons. The entry format is:

fieldname=value [*:fieldname=value*:...]

You must specify the login name field for each user in the batch file. Figure 95 illustrates an example nu batch file.

Figure 95 Example nu batch file

```
login=jwild:username=John Wild:office=100:extension=235
login=jdopey:username=James Dopey:office=250:extension=895
```

Step 5 Invoke nu in batch mode using the format

nu [-n *file_name*] -f *batch_file*

where

file_name is the name of the file that contains default constant values. If you do not specify the -n option, the system uses the default file /etc/nurc.

batch_file is the name of the file that contains user-specific information.

In batch mode, the default password will be the user's login name.

Step 6 Customize the working environment (.cshrc and .login files) for each individual user if necessary.

Adding users manually

Although you can add users manually, the preferred method is to use the nu utility, as nu verifies much of the user information before adding the new user. Adding users manually can induce errors.

Step 1 Log in as the superuser.

Step 2 If you are planning on adding users with a group ID that is not currently in the /etc/group file, add the new group to the /etc/group file before you start. Each line in this file represents one group; fields in this line are separated by colons. The format for an entry in the group file is:

group name:unused field:group ID:group members

where

group name is the name of the group, and is from 1 to 8 alphanumeric characters long.

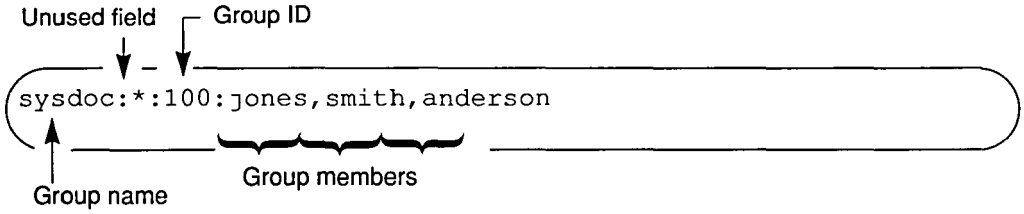
unused field is an unused field and must always contain an asterisk.

group ID is any unique number between 0 and 32767. When assigning GIDs, assign numbers in sequence. Do not use numbers 0 through 99 as these are reserved for use by CONVEX.

group members are individual users who are members of the group. Each user name included in the list is separated with a comma. A user can belong to as many as 16 groups.

Figure 96 shows an example entry in the /etc/group file.

Figure 96 Sample /etc/group entry



Step 3 Create an /etc/passwd entry using the `vipw` utility. The `vipw` utility uses the `vi` editor unless a different editor is specified in the VISUAL or EDITOR environment variable.

Caution

It is mandatory to use `vipw` rather than `vi` because `vipw` simultaneously changes both the /etc/passwd and /etc/pwrestrict files. It also makes certain checks to ensure that /etc/passwd contains only correct entries.

Only one user at a time can invoke `vipw`. When you invoke `vipw`, you edit a mixture of data from the /etc/password and /etc/pwrestrict files; changes are stored to both files. Include a line for each user you are adding to the /etc/passwd file. Each line represents one new user; fields in this line are separated by colons. The format of the entries presented by `vipw` is

name:password:UID:GID:comment:directory:shell:pwd_rest

where

name is the user's login name. The login name cannot have more than eight alphanumeric characters and by convention is lowercase. Do not use underscore, hyphen, or punctuation characters in the login name.

password is an encrypted password required for the user to log in. If you place an asterisk in this field, the user cannot log in from the local machine. (If someone created a home directory containing a viable .rhosts file for the user, the user could log in from remote machines.) Leave this field blank for now.

UID is the unique UID number. Do not use numbers 0 through 99, as these are reserved for use by CONVEX. The next number to use is maintained in the /etc/uidcount file. View that file using `cat` to get the next number.

GID is the user's default GID number. This number is either the system default group or one selected from the /etc/group file.

comment contains the following information, separated by commas. For historical reasons, this field is called the "GECOS" field.

```
real name, office number, phone ext,
home phone
```

These fields are optional. This information is used by utilities like `finger`, `mail`, and `news`. Use the ampersand symbol (&) in the real name field and the system will insert the login name. You can optionally enter this information using the `chfn` command after the user is added to the /etc/passwd file. See the chfn(1) man page for details on using this command.

directory is the user's login (or home) directory.

shell is the shell that is started after a successful login. If left blank, /bin/csh is used.

pwd_rest are the password restrictions that apply to the user. Contains the following information separated by commas.

min_wks,max_wks,pwd_date,pwd_req

min_wks is the minimum number of weeks the password is valid. The password must remain unchanged for this period of time.

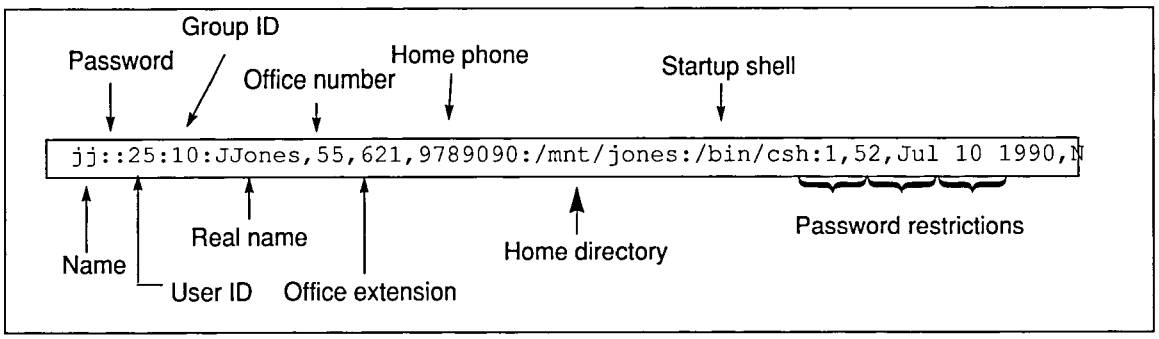
max_wks is the maximum number of weeks the password is valid. When the password is no longer valid, the user is prompted to set a new password.

pwd_date is the date the password was set. Passwords are aged from this date.

pwd_req specifies whether a password is required for the user to log in. `Y` requires a password; `N` does not.

Figure 97 shows an example line in a `vipw` session.

Figure 97 Sample `vipw` line



See the vipw(8) and passwd(1) man pages for details on using `vipw`.

Step 4 Save and close this file. The /etc/passwd file is updated. If the /etc/pwrestrict file exists, it is also updated, and the database is rebuilt.

Step 5 If the /etc/pwrestrict file does not already exist, create it using the `genrest` command. The `genrest` command creates an entry for each user in the /etc/pwrestrict file using the information available in the /etc/passwd file.

By default, no password restrictions are applied. To specify typing restrictions, use the `-t` option with the `genrest` command; to specify the minimum period in weeks that must pass before the password can be changed, use the `-m` option; and to specify the maximum number of weeks for which the password is valid, use the `-M` option. (The default minimum is 1 week; maximum is 52 weeks.) For example,

```
# genrest -t -mn15 -M30
```

creates a password restriction file that requires typing restrictions on the password. A minimum of 15 weeks must elapse before the password can be changed, and the password is valid for a maximum of 30 weeks.

If the password restriction file exists when you run the `genrest` command, you receive the error message:

```
genrest: /etc/pwrestrict already exists
```

For NFS sites, you can install the /etc/pwrestrict file as an NIS map on the network file server. Refer to the *ConvexOS Network File System System Manager's Guide* for details on setting this up.

Step 6 Add a password for each new user using `passwd`. `passwd` prompts for the new password twice, as shown in Figure 97.

Figure 98 Sample use of `passwd`

```
# passwd jones
New password:
Retype new password:
```

The `passwd` command updates both the /etc/passwd and /etc/pwrestrict files. If the password field for a user is left blank in the /etc/passwd file, no password is required to login. This would cause a security risk, so be sure to set an initial password for new accounts.

Step 7 Create a home directory for each new user. For example, enter:

```
# mkdir /mnt/jones
```

Step 8 Be sure the shell initialization files located in /usr/skel contain the commands and variables you want. You can modify these files using an editor.

Step 9 Copy the login shell initialization files in the /usr/skel directory in each new users home directory. For example, enter:

```
# cp /usr/skel/{.??*,*} /mnt/jones
```

Step 10 Change ownership of the login directory and its contents to the appropriate user using the `chall` command. For example,

```
# chall -o jones -g pubs /mnt/jones
```

changes all the files in directory jones to be owned by jones with a GID of pubs.

Step 11 Using an editor, increment the UID count in the /etc/uidcount file to reflect the next number to assign to a new user.

Step 12 Customize the working environment (.cshrc and .login files) for each individual user, if necessary.

Adding group membership

A user can belong to up to 16 groups. Membership in a group is specified in the /etc/group file. Each line in this file represents one group; fields in this line are separated by colons. The format for an entry in the group file is

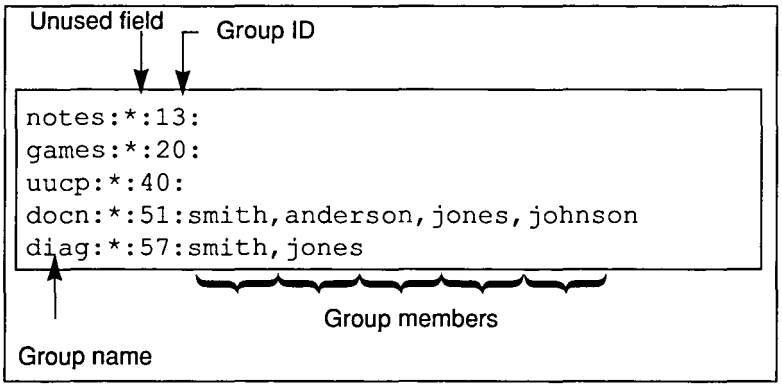
group name:unused field:group ID:group members

where

group name	is the name of the group from 1 to 8 alphanumeric characters long.
unused field	is an unused field and must always contain an asterisk.
group ID	is any unique number between 0 and 32767. When assigning GIDs, assign numbers in sequence. Do not use numbers 0 through 99 as these are reserved for use by CONVEX.
group members	are individual users who are members of the group. Each user name included in the list is separated with a comma.

An example /etc/group file is shown in Figure 99.

Figure 99 Sample /etc/group file



When a user creates a file, the new file inherits the group ownership of the directory in which the new file resides. The user's own groups have no bearing on the group owner for the newly created file.

Removing user accounts

When a user account is obsolete, you should remove the ability to log into that account, and optionally remove the files residing in the user's home directory.

Note

If disk space is not an issue, or if you wish to retain the files for whomever is taking over the user's projects, you only need to perform Step 1 to prohibit the user account from being used.

Use the following steps to deactivate and remove user accounts:

Step 1 Using the `vipw` utility, edit the /etc/passwd entry for the ex-user. Invoking `vipw` opens the /etc/passwd file.

Change the password for the user you are removing to something invalid, such as two asterisks (**). Also, change the user's login shell to /bin/false. See the vipw(8) man page for more details on using this utility.

Step 2 Create a directory in the /tmp directory for the user you are removing. For example, if you are removing user smith, enter:

```
# mkdir /tmp/smith
```

This is in preparation for collecting and removing all of user smith's files.

Step 3 Locate all files owned by the user using the `find` command. For example, the following command locates all the files for user smith and prints their names to a file called /tmp/smith/files:

```
# find / -user smith -print > /tmp/smith/files
```

Step 4 Archive the user's directories and files to tape. To make this easier, use /tmp/smith/files created in the previous step as input to the `cpio` command. For example,

```
# cat /tmp/smith/files | cpio -ocvB > \ /dev/rmt/0m
```

Step 5 Remove the user's home directory. For example, to remove user smith's home directory located on /mnt, enter:

```
# rm -fr /mnt/smith
```

If you must preserve the user's home directory, look for and remove any .crontab files in that directory.

Step 6 Review the list created in Step 3 and optionally remove any other files owned by the obsolete user that were not in the user's home directory.

Step 7 Check for and remove jobs queued in `cron` by this user.

Step 8 Remove the user's mail file. (In some cases you may wish to forward a user's mail. See the mail(1) man page for more information on mail forwarding.) For example, to remove user smith's mail file, enter:

```
# rm /usr/spool/mail/smith
```

Step 9 Using an editor, remove the user name from global mail aliases in the /usr/lib/aliases file.

Step 10 Using an editor, remove the user from any groups in the /etc/group file.

Step 11 Use the `edquota` command to set the user's limits to zero. Refer to Chapter 9, "Setting quotas on disk space use," on page 191, for details on how to do this.

Step 12 If the user knew the root password or any dial-up passwords, change those passwords.

Setting up the accounting system

8

The accounting system keeps track of the system resources an individual user or group uses. It collects the following types of information:

- Process terminations
- Login times
- Successful and unsuccessful changes in billing accounts
- Tape allocations and deallocations
- Tape error messages
- Printer use
- Printer use errors

From information collected through the accounting system, you can determine how much disk space, line printer and tape use, or computer 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
- Effectively manage system resources
- Keep strict accounting of project costs, because users can bill resource use to specific projects

This chapter discusses how to set up the files required by the accounting system. For information on how to generate reports using the accounting information collected, see the *Operations Guide*. You must be superuser to perform the tasks described in this chapter.

How accounting works

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.

When a user logs in, the accounting system automatically begins collecting accounting information for the user's default billing account. The default billing account for each user is the group specified for that user in the /etc/passwd file and activity code zero, a miscellaneous account, unless a different account is specified in the user's .login file using the `bill` command. (The user must be using the C-shell in order for you to use the .login file to assign a default billing account.) The format for the `bill` command is:

`bill` *group_id activity_id*

Users can use `bill` from the command line to change their billing account at any time.

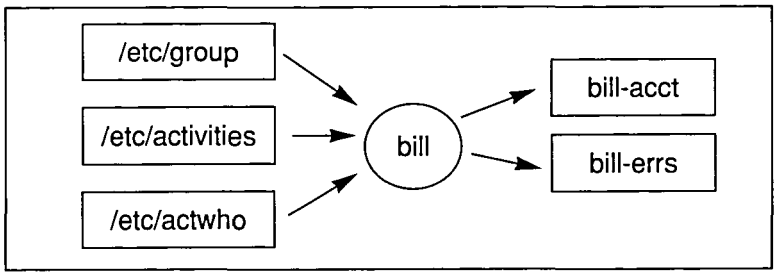
In addition to getting the default billing account for /etc/passed, `bill` gets information from three user-generated files. These files and their purposes are listed in Table 21.

Table 21 Files used by the `bill` command

Files	Purpose
/etc/group	Defines valid billing groups
/etc/activities	Defines valid billing activities
/etc/actwho	Defines which user and groups can bill to a particular activity. `bill` checks the /etc/actwho file before changing a user's account to determine whether or not a user is allowed to bill to the specified account.

The `bill` command uses these files to generate entries to the bill log file named bill-acct located in the /usr/adm directory. The input and output for the `bill` command is illustrated in Figure 100.

Figure 100 Input and output for the `bill` command



If the input file (that is, group, activities, and actwho) are not set up, accounting always uses the group specified for the user in the /etc/passwd file and an activity code of zero.

When a process forks a child process, the current billing account is passed to any child processes generated. The current billing account for a process is stored in the kernel.

Collection log files

Accounting data is collected in log files located in the /usr./adm directory. In most cases, the file only needs to exist in order to have the information collected. Following is a list of files that, if they exist in /usr/adm, automatically collect the specified information once the system boots to multi-user mode:

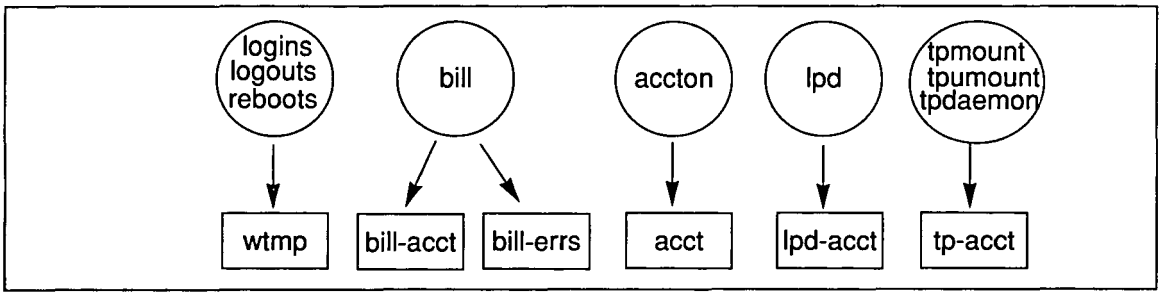
- wtmp—Collects login, logout, and reboot activity generated by login, logout, and reboot utilities.
- bill_acct—Collects successful billing account changes generated by the bill utility.
- bill_errs—Collects unsuccessful billing account changes generated by the bill utility.
- tp_acct—Collects tape use data generated by the `tpmount` and `tpunmount` utilities, and `tpdaemon`.

There are two collection cases that require more files than those that exist in /usr/adm. These files, their purposes, and the additional requirements are listed below.

- lpd_acct—Collects printer use data generated by `lpd`. To collect this information, the file must exist and the file name must be defined in the af option of the /etc/printcap file for the pertinent printer(s). Refer to Chapter 5, "Setting up the line printer system," on page 129, for details on how to do this.
- acct—Collects process terminations. To collect this information, the file must exist and accounting must be turned on using the `accton` utility.

This relationship between the collection files and the utility that generates the information is illustrated in Figure 101.

Figure 101 Input for accounting log files



Setting up accounting files

You must perform the following steps to set up the files required by the accounting system:

Step 1 Log in as the superuser.

Step 2 Using an editor, add any billing groups you want in the /etc/group file, if they do not already exist. To speed bill access time, list entries in this file so the most-often-used groups are first.

Each line in this file represents one group; fields in this line are separated by colons. The format for an entry in the group file is:

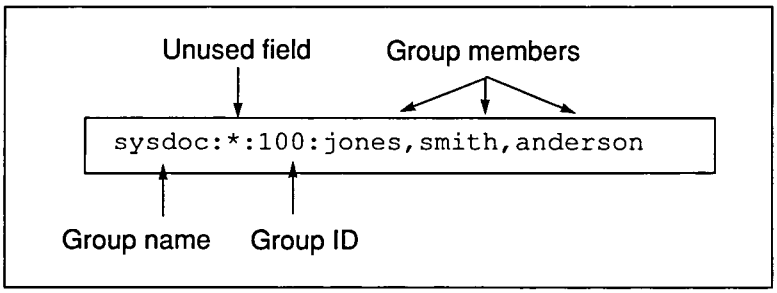
group name:unused field:group ID:group members

where

group name	is the name of the group from 1 to 8 alphanumeric characters long.
unused field	is an unused field and must always contain an asterisk.
group ID	is any unique number between 0 and 32767. When assigning GIDs, assign numbers in sequence. Do not use numbers 0 through 99 as these are reserved for use by CONVEX.
group members	are individual users that are members of the group. Each user name included in the list is separated with a comma. A user can belong to as many as 16 groups.

Figure 102 shows an example of an entry in the /etc/group file.

Figure 102 Entry in the /etc/group file



Step 3 Enter the group ID number in the `Group id` field. Do not assign a group ID number between 0 and 99, because these are reserved for use by CONVEX. The /etc/group file is checked to be sure the group ID entered is unique before it is added to the /etc/groups file.

Step 4 Using an editor, add any activities you want to bill for in the /etc/activities file. To speed `bill` access time, list entries in this file so the most-often-used activities are first.

Figure 103 illustrates an example activities file.

Figure 103 Eample /etc/activities file

```
misc:0
backup:18
random:38
develop:58
support:78
```

Each line in this file represents one activity; fields in this line are separated by a colon. The format of an entry in the activities file is

name:number

where

name is the name of the activity.

number is the unique ID number associated with the activity required by the bill utility. This number is used when generating accounting reports.

If you have CXbatch configured on your system, it is important that you do not assign consecutive numbers when you assign numbers to billing activities. Instead, assign them in increments of 10 or 100. The amount of incremental space between numbers affects the activity IDs assigned to jobs by CXbatch. If you assign numbers without incremental space between them, it is difficult to trace origin of jobs. See the *CONVEX CXbatch System Manager's Guide* for details.

Step 5 Be sure to set up an activity 0 (zero) called overhead or miscellaneous as a default activity.

Step 6 Change the permissions on the /etc/activities file to permit read and write access to owner, and read access to group and other. To do this, enter:

```
# chmod 644 /etc/activities
```

Step 7 Using the `edactwho` command, define which users and groups can bill to a particular activity. Enter:

```
# edactwho
```

Step 8 The edactwho utility invokes the shell's default editor and opens a temporary file. After editing is completed, the edactwho utility checks the new file for correct syntax, then moves its contents to the /etc/actwho file. Figure 104 illustrates an example actwho file.

Figure 104 Example /etc/actwho file

```
*.misc:*
lp.*:george,larson
engineer.develop:smith,george
engineer.support:williams,george
project1.design: jones,smith
project2.document:johnson,smith,anderson
project?.*:brown,jones
```

Each line in this file represents one group/activity combination (or billing account), and defines the users that can use that combination for billing purposes. The format for an entry in the /etc/actwho file is

group.activity:user[,*user*,....]

where

group is a group specified in the /etc/group file.

activity is an activity specified in the /etc/activities file.

user is a single user or list of users allowed to bill to this group/activity. User names included in this list are separated by commas.

You can use the all-character wildcard (*) in any of these fields. Used in the group field, all groups are valid for the activity. Used in the activity field, all activities are valid for the group. Used in the user field, all users can bill to the group/activity.

You can use the single-character wildcard (?) in the group or activity field. Any group or activity that matches is valid for the entry.

In the example shown in Figure 104, the asterisk (*) in the group field on the first line specifies all groups are valid; the asterisk in the user field on the first line specifies all users are valid. The asterisk in the second line specifies all activities are valid. The question mark (?) in the last line specifies that any group beginning with project followed by a single character is valid. See the edactwho(8) man page for more information on this file.

Step 9 Change the permissions on the /etc/actwho file to permit read and write access to owner, and read access to group and other. To do this, enter:

```
# chmod 644 /etc/actwho
```

Step 10 Change your working directory to /usr/adm by entering:

```
# cd /usr/adm
```

Step 11 Create the desired log files in this directory using the `touch` command. The format is

`touch` *filename* [*filename* ...]

where *filename* is the name of the file you want to create. This can be one or more of the following:

`wtmp`	Collects login, logout, and reboot activity.
`bill-acct`	Collects successful billing account changes.
`bill-errs`	Collects unsuccessful billing account changes.
`tp-acct`	Collects tape use data.
`lpd-acct`	Collects printer use data. (This file name is typically used to collect printer use data for all printers. However, if you want to keep separate accounting information for each printer, you must create a separate log file with a unique name for each printer.)
`acct`	Collects process terminations. This file, unlike the others, is activated with the `accton` command.
`savacct`	Collects process termination summary. Must be activated with the `accton` command.
`usracct`	Collects process termination summary by user and group. Must be activated with the `accton` command.

Step 12 Change the mode on each of the files created in Step 11 to permit read and write access to owner, and read access to group and other. To do this, enter:

```
# chmod 644 filename [filename...]
```

Step 13 If you created a single log file or multiple log files in Step 11 to collect printer use data, modify the /etc/printcap file using an editor, to contain the name of the log file that stores accounting information for each printer entry in the /etc/printcap file. This information is specified with the `af` variable. For example, add the following line:

```
:af=/usr/adm/lpd-acct
```

If you want to keep separate accounting information for each printer, the log file name can be different for each printer. However, be sure to create an empty log file in the /usr/adm directory for every name specified in this file (see Step 11 and Step 12). An example printer entry in the /etc/printcap file is shown in Figure 105.

Figure 105 Example /etc/printcap entry

```
swip |Imagen in software area\
          :lp=/dev/null:\
          :rt=/usr/spool/lpd/swip/resfonts:\
          :hn=swip:\
          :af=/usr/adm/lpd-acct:          <-------  Specifies log file
```

Step 14 If you do not want the default accounting for users to be by the group specified in the /etc/passwd file with an activity code 0 (miscellaneous), and if your users are using C-shell, put the default billing account for each user in their .login file using the trusted editor, `ed`.

For example, placing the following line in a user's .login file will set the default billing account to group project1, activity code design.

```
bill project1 design
```

Step 15 If you created an acct file in Step 11, execute the `accton` command to start collecting system accounting information to the acct files for each process executed. You must specify the acct file in which to store the collected information. The acct files are:

acct	For raw, per-process accounting
savacct	For per-process summary
usracct	For per-user and per-group activity summary

To turn any of these accounting files on, enter:

```
# accton /usr/adm/acct
```

To turn off accounting to these files, use the `accton` command without arguments.

When disk space on the partition holding the /usr/adm directory exceeds 98% capacity, the system suspends logging. The system automatically restarts logging when space drops to 96%.

Step 16 Using an editor, place the `accton` command in the /etc/rc.local file to automatically start accounting when the system is booted. You must specify the file in which to store the collected information. Enter the following line in the /etc/rc.local file:

```
accton /usr/adm/acct
```

Step 17 If you have purchased CXbatch, an optional product, accounting requires some set up in qmgr. To enable batch accounting, enter the following lines in the qmgr file for each queue:

```
set activity_id_offset=0-9

set aid_mask=increment from /etc/activities

set accounting=on

set acc_logfile /usr/adm/batch-acct
```

Refer to the *CONVEX CXbatch System Manager's Guide* for the procedure for enabling batch accounting.

Setting quotas on disk space use

9

The disk quota system is an optional feature that provides a mechanism to control use of disk space by users. You can set quotas using the `edquota` command for any or all users on any or all file systems.

Using the `edquota` command, you can restrict the amount of disk space a user can use, the number of files (inodes) a user can own, or both. If both limits are set, the user is restricted by whichever limit is exceeded first.

You can set soft and hard limits for both the amount of disk space and the number of files. Once a user exceeds the soft limit, the system issues a warning message to the user's terminal but allows the user to continue working for a limited amount of time. This time limit is also set for the user with the `edquota -t` command.

Once the time limit or hard limit is reached, the user is not allowed to write any more data to the disk. That is, requests for space or attempts to create a file fail. Hard limits and time limits cannot be exceeded.

On the first failure, the system issues a message to the user's terminal. Only one message is sent each time a hard limit is reached, no matter how many failures occur. The only way users can reset this condition is to reduce disk use below the specified quotas. The system manager can reset this condition by increasing the user's quota limits or turning quotas off.

Users modifying a file owned by someone else are subject to the quota limit of the owner of that file. When the soft limit is exceeded, the error message is issued to the owner of the file, and the hard quota time limit countdown begins for the owner of the file. Occupied disk space must be reduced below the limit to reset the condition.

This also applies to users extending files owned by root. If the user root quota limit is set to zero, which specifies no quota limits apply, all users with write permissions to files owned by root are allowed to write until the partition fills up, regardless of personal quota limits. Because of this, you must decide whether or not to place quota limits on the user root.

No warnings or error messages are printed if a user exceeds soft or hard limits on an NFS-mounted file system. If this is likely to cause problems at your site, instruct users running jobs on remote systems to run `quota` on the remote system before starting a job.

Disk quota use and limits are stored in a file named quotas located on the mount point of the file system where the quotas are imposed. The data in the quotas file is an array of structures, indexed by UID, with one structure for each user on the system regardless of whether the user has a quota on the file system. Do not change this file name because several user-level utilities depend on it. Also, do not copy the quotas file.

Use the following procedure to set quotas for users and file systems:

Step 1 Determine which file systems require quotas. Usually, only file systems containing user home directories or other user files need quotas. If possible, the /tmp file system should not have quotas.

Step 2 Decide on the block or inode limits for each user in the file system where you will implement quotas. You can impose any combination of hard and soft limits for each user, and limits can be different for each user, but typically set block or inode limits, not both. Always specify a time limit.

A limit set to zero is disabled. Disabling all limits for a user disables the entire quota for that user; that is, no quotas are imposed.

Step 3 Log in as the superuser.

Step 4 All quota administration commands must be run on mounted file systems. Use the `df` command to check that the file systems to receive quotas are mounted. Enter:

```
# df
```

Example output for this command is shown in Figure 106. If the file system appears in this output, it is currently mounted.

Figure 106 Example `df` output

```
% df
Filesystem     kbytes    used     avail  capacity   Mounted on
/dev/da0a      20143     17639    489    97%        /
/dev/dd0h      261215    85615    149478 36%        /doc
/dev/dd0g      401439    218906   142389 61%        /usr
/dev/dd0a      44159     20017    19726  50%        /usr/adm
/dev/da0g      183887    122253   43245  74%        /mnt
```

Step 5 If the file system is not mounted, mount it using the `mount` command. For example, to mount the /mnt partition, enter:

```
# mount /mnt
```

Step 6 Create a file called quotas in each file system that will maintain quotas. For example, to create a file called quotas in the /mnt and /doc file systems, enter:

```
# touch /mnt/quotas /doc/quotas
```

Step 7 Set quota limits using the `edquota` command. You can do this with a command line or interactively. For example, the following command line sets a soft block limit of 100 and a hard block limit of 120 for user smith and jones on the /mnt file system:

```
# edquota -b 100 -B 120 /mnt smith jones
```

To change limits interactively for an individual user, do not specify limits on the `edquota` command line. For example, enter:

```
# edquota smith
```

The information shown in Figure 107 is displayed using the default editor. Edit the limits, save, and close the file.

Figure 107 Example `edquota` interactive file

```
        ---> Quota information for smith<---
fs /mnt blocks (soft = 100, hard = 120) inodes (soft = 0, hard = 0)
            current blocks = 0; current inodes = 0
```

Step 8 Set the time limit for the file systems using the `edquota` command. A single time limit applies to all file systems. Enter :

```
# edquota -t
```

The information shown in Figure 108 is displayed using the default editor. Edit the time limit, save and close the file.

Figure 108 Example `edquota -t` interactive file

```
fs /mnt blocks time limit = 0 (default), files time limit = 2
                    day(default)
```

You can set the time in seconds by specifying sec, minutes by specifying min, hours by specifying hour, days by specifying day, weeks by specifying week, or months by specifying month.

After setting the quotas for a user, you can use their record as a prototype to duplicate quotas for other users by using the `-p` option. For example, the following command duplicates the quotas set for user pat for users chris and francis:

```
# edquota -p pat chris francis
```

See the edquota(8) man page for more information on each of these options.

Step 9 Initialize the newly created quotas using the `quotacheck` command. For example, to initialize the quotas on the /mnt file system, enter:

```
# quotacheck /mnt
```

Step 10 Enable the quota system using the `quotaon` command. For example, to enable the quota system for the /mnt file system, enter:

```
# quotaon /mnt
```

Quotas are then enforced for the specified file system. You can enable the quota system for more than one file system by separating the file system names with a blank space.

Step 11 Check to be sure the following lines exist in the /etc/rc.local file. These lines automatically enable the quota system when the system is booted.

```
quotacheck -p -a

quotaon -a
```

Step 12 If these lines do not exist in the /etc/rc.local file, add them.

Step 13 If you did not perform Step 11, skip to Step 15. If you performed Step 11, check to be sure the file systems you want started on system boot have their read, write, and quota options set in the /etc/fstab file. To do this, enter:

```
# less /etc/fstab
```

The output from this command is shown in Figure 109. Column 4 indicates the read, write and quota option settings.

Figure 109 Example /etc/fstab listing

```
/dev/da0a     /            4.2       rw         1     1
/dev/da0b     swap_1of2    ignore    rw         0     0
/dev/da0g     /mnt         4.2       rw,quota   2     3
/dev/da1a     /usr/spool   4.2       rw         1     2
/dev/da1b     swap_2of2    swap      rw         0     0
/dev/da1d     UNUSED       ignore    xx         0     0
```

Read, write, and quota options specified here

Step 14 If the read, write, and quota options are not set, edit the /etc/fstab file so they are set.

Step 15 To turn the quota system on for NFS clients, add the following two lines to the /etc/rc.local file on the NFS server machine:

```
quotacheck -a
quotaon -a
```

Quotas on remotely mounted file systems work much as quotas on locally mounted file systems, except that no warnings are printed when the user exceeds the soft limit. Your NFS users should periodically use the `quota` command to find out the state of their quota allocation. For more information on NFS, see the *CONVEX NFS System Manager's Guide* for details.

Setting up sendmail 10

This chapter presents information on setting up sendmail, the internetwork mail router. The major topics in this chapter are:

- Understanding sendmail
- Using aliases
- The sendmail configuration file
- Modifying the sendmail configuration file
- Testing the configuration

For information on sendmail operation and maintenance, refer to *Managing ConvexOS: Operations Guide*, the chapter "Managing sendmail."

Understanding sendmail

sendmail is a highly-configurable message routing facility. It does not interface directly with the user or perform actual mail delivery, but relays incoming and outgoing mail messages to the appropriate programs for delivery or further routing. sendmail can route messages over a local area network or through a gateway and can be configured to work with several transport protocols.

The domain naming system allows interoperation among heterogeneous systems on the Internet. Consequently, a destination address in a message header may not be understood by the transport system of the originator. An example of this is the combination of *user@domain* and UUCP *host!user* style addressing. A major function of sendmail is to convert message formats between disparate networks.

In addition, sendmail has the following capabilities:

- Message queuing
- Error handling
- Automatic routing to network gateways
- User-controlled addressing through mail forwarding and mailing lists
- System-wide aliasing
- Access to Berkeley Internet Name Domain server (BIND) and external name server programs
- Simple Mail Transfer Protocol (SMTP) server

Before you begin setting up sendmail, it is helpful to understand how it works and what it can do. The following sections explain sendmail operation and capabilities.

How sendmail works

sendmail handles mail messages in two distinct phases. In the first phase, it collects and stores messages from any of the following sources:

- A User Agent (UA), a user program such as mail, elm, or xmh that is used to create, read, and manage messages
- A user that invokes sendmail directly
- A receiving agent (such as the sendmail daemon) that calls sendmail to route incoming messages
- The mail queue

In the second phase, the message is routed. To route a message, `sendmail`:

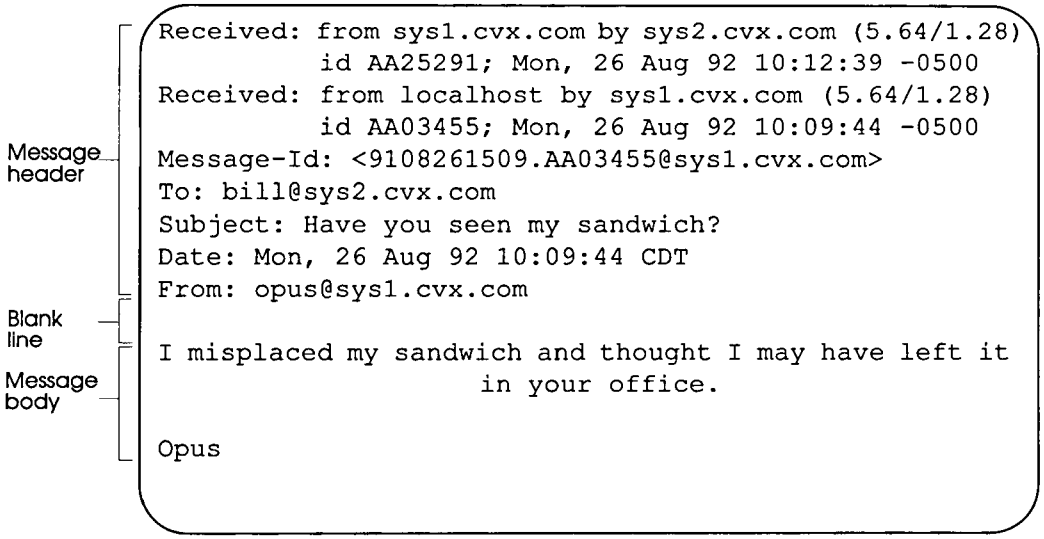
- Rewrites the recipient and sender addresses to the form required by the target network
- Chooses a mailer, often based on the addresses inside the message. The mailer is also known as a Message Transfer Agent (MTA).
- Reformats the header as required
- Passes the message to the mailer

A message has three parts:

- **Envelope**—Contains routing information used by programs that create, route, and deliver the message. The envelope is not visible to the sender or recipient of the message. Some of the envelope information, such as the sender and recipient address, is also in the message header.
- **Message header**—A series of text lines consisting of a field designator (for example, `To:`, `From:`, `Date:`, `Subject:`) followed by the appropriate information. Standards for message headers are defined in Request for Comments (RFC) 822.
- **Message body**—The text that is passed from the sender to the recipient. The message body begins after the first blank line in the message following the message header.

Figure 110 shows an example of an incoming mail message.

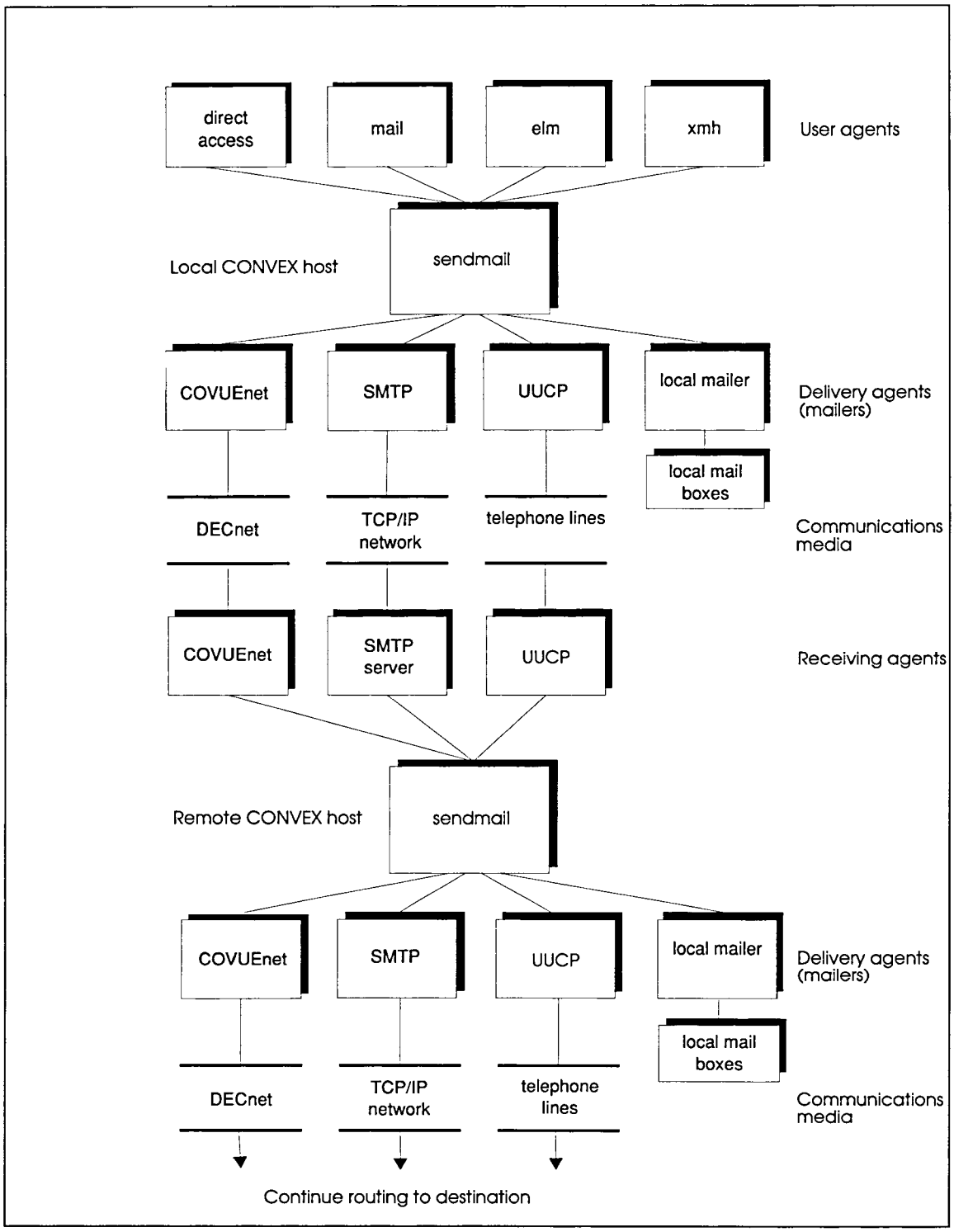
Figure 110 Sample mail message



This mail message illustrates what the recipient sees. The envelope is not visible.

Figure 111 shows how messages to both local and remote destinations on CONVEX systems flow through `sendmail`.

Figure 111 Flow of messages through `sendmail`



Routing messages

After `sendmail` collects a message, it first parses the recipient address on the message envelope to determine the transport mechanism and the next step in the chain of mail relays. `sendmail` applies rules defined in the `sendmail` configuration file, /usr/lib/sendmail.cf, to resolve the recipient address to the format:

{*mailer, host, user_address*}

Routing to local destinations

If `sendmail` determines that the destination is the local system, it looks up the address in the aliases database and .forward files, if any, to expand the address if necessary. Aliasing can be used to route mail to programs and to files as well as to multiple recipients. (Refer to "Using aliases," on page 204.)

After expanding the alias, `sendmail` routes mail addressed to a local user to the local mailer (/bin/mail) that deposits the mail in the user's mailbox.

Routing to remote systems

If the destination is a remote system, `sendmail` invokes one of the following mailers based on rules in the sendmail.cf file:

- UUCP: uux/rmail
- COVUE: cumail/cumaild
- SMTP

`sendmail` invokes each mailer separately for each host to which it routes mail. Some mailers can be invoked once to route to multiple recipients on a given system; others must be invoked separately for each recipient.

`sendmail` invokes a mailer by creating a pipe and forking a process to execute the mailer with a command line specified in the sendmail.cf file. The parent process writes the message header and body to the pipe. `sendmail` passes the recipient address on the command line to the mailer or opens a bidirectional pipe to the child process and transmits the message using SMTP.

If a destination address resolves to a mailer that delivers messages over Ethernet, `sendmail` opens a TCP/IP connection to the SMTP server on the specified host and transmits the message using SMTP.

The mail queue

Messages that are temporarily undeliverable are saved in a mail queue for later delivery. The message body and the header are saved in one file, and the envelope is saved in another file in the mail queue directory /usr/spool/mqueue. `sendmail` normally processes the queue at specified intervals. You can force the queue to be processed immediately by invoking sendmail with the `-q` flag and no specified interval. If the message is still undeliverable after a specified amount of time (the default is three days), it is returned to the sender.

Messages can also be queued when the system load average reaches the limit specified by the `x` option in the sendmail.cf file.

Error handling

In addition to queuing messages that are temporarily undeliverable, `sendmail` handles permanent failures such as an unresolvable recipient address or the inability to find the Internet address of a remote host. If a message is permanently undeliverable, `sendmail` returns the message to the sender with a transcript of the failed delivery attempt. Any standard error output from the mailer that failed is included in the transcript.

Using aliases

An alias is an alternate name for a recipient or a list of recipients. Aliases are used to:

- Assign additional names to a user
- Forward a user's mail to a particular machine
- Define mailing lists (a group of users)
- Send mail to a file or program

Aliasing occurs only on addresses that resolve to local names. When routing mail, `sendmail` looks up local addresses in the alias database. If a local address is listed there, `sendmail` expands it to the address or list of addresses specified. If the alias database files do not exist, or if `sendmail` is run with the `-n` flag, aliasing is not done.

System-wide aliases are maintained in the text file /usr/lib/aliases and the dbm format files /usr/lib/aliases.dir and /usr/lib/aliases.pag. The dbm files are built from the text file when you start `sendmail` (if the `D` option is specified in the configuration file) or when you execute the `newaliases` command. The dbm files speed up searching the database.

System-wide aliases are used for permanent routing changes. Users can make temporary routing changes with a .forward file located in their home directory.

This section explains how to set up the aliases database and use the .forward file, and describes configuration options you can use to control aliasing.

Creating the alias database

To create the alias database, add lines of the following format to the /usr/lib/aliases file:

alias: name1, name2, . . .

alias is how mail can be addressed, and the name list is the list of addresses, separated by commas, to which mail is to be forwarded. Name can be:

- A local user name
- A remote address
- Another alias
- An absolute path name
- A file containing a mailing list

To continue an entry onto the next line, begin the line with a space or tab. Lines beginning with a pound sign (#) are comments. Blank lines are ignored.

Local user names in the name list that are preceded by a backslash (\) will not be expanded further.

The following are example entries from the /usr/lib/aliases file:

```
cat: bill

bill: bill@ack

dev: bill, opus, bloom!milo
```

The first entry indicates that mail sent to cat should be forwarded to bill; the second line causes mail addressed to bill to go to bill on the system ack; and the third line is a mailing list that sends mail addressed to dev to bill, opus, and bloom at milo.

The syntax and requirements for file name, command line, and include file aliases is described below.

File name aliases

If the destination of an alias is a file, the file must already exist, be specified by an absolute path name, not be executable, and be readable and writable by all users unless the `sendmail` daemon (started by invoking `sendmail` with the `-bd` flag) is running with setuid to root. `sendmail` appends the message to the file.

The format of a file name alias is

alias: *file_pathname*

where *alias* is the alias name, and *file_pathname* is the absolute path name of the file that `sendmail` appends messages to. The first / tells `sendmail` that the recipient is a file rather than a login name.

The following is an example of a file name alias:

```
status: /mnt/project/status
```

Command aliases

You can also have aliases for commands. When you alias to a command, the command gets executed by the `prog` mailer specified in the configuration file. The command is executed as either the user executing `sendmail`, or the user and group set with the `u` and `g` `sendmail` configuration file options.

The format of a command alias is

alias: "| *file_pathname*"

where *alias* is the alias name and *file_pathname* is the absolute path name of the command or program. The pipe (|) character informs `sendmail` that the rest of the address is to be processed as a shell command; `sendmail` then pipes the message as standard input to the specified command. The double quotes are necessary to protect blanks in the command line.

If the exit status of the command is nonzero, output to stderr is included in the error transcript that `sendmail` returns. Otherwise, the stdout and stderr of a command must be redirected to preserve it.

Aliasing to a command can be used to implement a news facility, collect statistics from remote systems, or respond to mail to a defunct user. For example, the entry

```
misc: "|/usr/local/bin/mail2news -n abc.misc"
```

sends mail to a news program.

Inclusion

Inclusion directs `sendmail` to read a file for a list of addresses. This is useful when you want to have a set of aliases that are local to a particular machine yet have a shared aliases file in a networked environment, or to enable users to maintain their own mailing lists without giving everyone write permission to the alias database.

The format of an inclusion entry is:

alias: `:include:` *file_pathname*

where *alias* is the alias name, and *file_pathname* is the absolute path name of the file containing a list of user addresses or additional aliases.

For example,

```
project: :include:/usr/project/userlist
```

instructs `sendmail` to forward all mail to the alias project to each of the users listed in the file userlist.

Special aliases

The following special aliases are either required or typically included in the /usr/lib/aliases file.

Postmaster

The aliases file is required by RFC 822 to have an entry for Postmaster. The Postmaster alias must resolve to the person responsible for the site's mail system.

MAILER-DAEMON

MAILER-DAEMON is the name of the sender of error messages and is defined by the `n` macro in the configuration file (refer to Table 23). MAILER-DAEMON is usually aliased to Postmaster.

List owner

When creating a mailing list, you can specify an owner of the list. If an error occurs when sending to a particular address, `sendmail` searches the alias database for an alias of the form `owner-`*alias* to receive the errors.

The format of the owner alias is:

`owner-`*alias*: *owner*

alias is the name of the list, and *owner* is the login name of the list owner.

In the following example

```
dev: bill, opus, milo
owner-dev: postmaster
```

if milo no longer had a mailbox on the system, "postmaster" would receive an error message.

List request

The list request alias is used to request that a name be added, removed, or changed in a mailing list. The format of the list request alias is:

list`-request`: *owner*

list is the list name and *owner* is the person responsible for maintaining the list.

For example:

```
dev-request: owner-dev
```

Mail sent to `dev-request` would be received by the owner of the dev alias.

nobody

The nobody alias is often aliased to /dev/null and is used for testing purposes.

Avoiding aliasing loops

Avoid creating aliasing loops, which can occur either locally or remotely.

Local loops

An example of a local aliasing loop is:

```
jane: jpc

jpc: jane
```

In this loop, mail addressed to jane expands to jpc, and mail addressed to jpc expands to jane. If mail addressed to several recipients includes recipients in a local aliasing loop, the recipients in the loop will not receive mail, but the others will.

If the only recipients for the message are in a local alias loop, the message is returned to the sender with the error message

```
All recipients suppressed.
```

Remote loops

In the following remote aliasing loop, the local system, kanga, has the following in the aliases file:

```
jane: jane@roo
```

The remote system, roo, has

```
jane: jane@kanga
```

in the aliases file, or user jane could have the entry

```
jane@kanga
```

in her .forward file on the remote system (the .forward file is described in the next section).

In this situation, mail will bounce between the two systems for a specified number of hops before `sendmail` returns the mail to the sender with an error message.

User-controlled aliasing

A user can specify addresses in a .forward file located in the user's home directory. `sendmail` looks for the presence of a .forward file, and if found, forwards the user's mail to the addresses specified in the file. The .forward file is useful for temporary mail routing changes.

The syntax for the .forward file is

```
user_address1, user_address2,...
```

where *user_address* is one or more addresses separated by commas or new line characters. Each address must have the same syntax as an address on the right-hand side of an alias expansion.

For example, if user opus on the machine homesys has the following in his .forward file

```
opus@devsys
```

mail sent to `opus@homesys` is redirected to opus' account on devsys. If the entry in the .forward file on homesys is

```
\opus@devsys
```

the backslash instructs `sendmail` to deposit the mail in the user's mailbox before redirecting it.

If two users have identical .forward files and both users are on the distribution list for a message, only the first user on the list receives a copy of the message. The contents of each .forward file must be unique because `sendmail` resolves addresses to the contents of the .forward files and then removes duplicates.

Aliasing configuration options

Five `sendmail` configuration file options affect aliasing or the aliases database. These options are described below. For information on specifying options in the `sendmail` configuration file, refer to the section "The `sendmail` configuration file."

Include sender in alias expansion

Normally, the sender is not included in alias expansions. For example, if milo sends mail to the dev alias and the expansion of dev includes milo, the message is not delivered to milo. Use the `m` option in the `sendmail` configuration file to include the sender in the alias expansion.

Rebuild database automatically

If you specify the `D` option in the configuration file, or specify the `-oD` flag on the command line, `sendmail` rebuilds the aliases database automatically if the date that /usr/lib/aliases was last modified is more recent than the date that the aliases database was last rebuilt.

In addition to specifying the `D` option, the aliases database files usr/lib/aliases.dir and /usr/lib/aliases.pag must be writable by all (mode 666), or `sendmail` must be running setuid to root for the aliases database to be rebuilt automatically.

If you do not set this option, you must use the `newaliases` command (`sendmail -bi`) to rebuild the database.

Problems can occur when more than one process tries to access the database; this is explained in the "Operating `sendmail`" chapter in *Managing ConvexOS: Operations Guide.*

Resolve right-hand side of alias definitions

If the `n` configuration option is set to `TRUE` when `sendmail` rebuilds the aliases database, `sendmail` attempts to resolve all addresses on the right-hand side of alias definitions (this is the default). If it is unable to resolve any of the addresses, `sendmail` reports an error for each unresolved address. It is preferable for `sendmail` to find errors when the database is being built, rather than when someone tries to mail to the alias; however, a large database can take a long time to rebuild.

You can prevent `sendmail` from resolving the right-hand sides of aliases at build time with the line `OnFALSE` in the configuration file (or with the `-onFALSE` option on the command line). Disabling this option can be useful when only a small part of the aliases database changed since it was last rebuilt.

Use a different alias database file name

The `A` option is used to specify a file name other than the default (/usr/lib/aliases) for the aliases database. If `A` is specified with no file, a file named aliases in the current directory is used.

Use the NIS aliases map

If you are running Network Information Services (NIS), use the `p` option to inform `sendmail` to query the NIS aliases map for alias entries. The NIS aliases map contains the same type of information as /usr/lib/aliases.

The sendmail configuration file

The behavior of the sendmail program is controlled by the configuration file /usr/lib/sendmail.cf. The configuration file:

- Defines certain names and formats (for example, the official host name and default header field formats)
- Sets values of operational parameters (for example, logging level and timeout values)
- Specifies how to interpret recipient addresses
- Specifies the mailer to use for message delivery
- Specifies how to rewrite header address information in a format understood by the receiving host
- The configuration file is organized as a series of different types of lines that can be grouped into three general categories:
  - Definitions of symbols, classes, options, and parameters
  - Address rewriting rules
  - Definitions of mailers and delivery programs, and how to invoke them

The first character of each line identifies the line type. Lines beginning with a space or tab are continuation lines. Lines beginning with a "#" are comments. Table 22 shows the line types and what they define.

Table 22 Syntax definition characters for sendmail.cf

Character	Defines
C or F	Classes
D	Macros
O	Options
P	Precedence classes for messages
T	Trusted users
H	Header line formats
S	Rule sets
R	Rewriting rules

The syntax for each of the line types is explained in the following sections.

C and F: classes

C and F lines define variables for classes of elements, such as all the machines of a particular type on the network. Classes are used in the address rewriting process to apply rewriting rules to all elements of a particular type. Classes can be defined directly in the configuration file or read in from another file.

Define a class in the configuration file

The syntax for defining a class directly in the configuration file is

C*c*member1 *member2...*

where

c is a single character for the class name.

member1 is the list of elements in the class separated by spaces. Use only uppercase letters for your site-specific class names because sendmail reserves all other characters for its internal use. Class members can also be defined on separate lines. For example,

```
CVmercury venus earth
```

and

```
CVmercury
CVvenus
CVearth
```

define class V to contain mercury, venus, and earth.

The special class w is set to all the names that the local host is known by (w is the macro defined by sendmail for the local host name).

Define a class from a file

The syntax for defining a class from a file is

F*c*file [*format*]

where

c is the uppercase single-character name of the file class.

fil is the absolute path name of the file containing the list of class members.

format is a `scanf` format that must produce a single string from each line in the file. `%s` is used by default if no format is specified. Refer to the scanf(3S) man page for more information on `scanf`.

For example, the definition

```
FU/usr/lib/uucp/L.sys %[abcdefghijklmnopqrs\
tuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ123456789\
0_-]
```

defines the class U, the UUCP hosts that this system talks to, as the contents of the L.sys file read with a `scanf` format that picks up only the host names. The format string is broken in the example to fit on the page, but should be a single line in the configuration file.

This method of defining the UUCP hosts eliminates the need to update the configuration file when UUCP hosts are added. However, if changes are made to the L.sys file, the `sendmail` configuration file must be refrozen. The frozen configuration file is an image of `sendmail`'s data space after reading in the configuration file. Refer to *Managing ConvexOS: Operations Guide*, the chapter "Managing `sendmail`" for more information.

D: macros

Macros define variables such as symbols, options, and other configuration parameters. Macro definitions begin with the letter "`D`" followed by a single character name. `sendmail` uses most of the lowercase letters for predefined macros that it uses internally; user-defined macros should be uppercase.

The syntax to define a macro is

`D`*xvalue*

where *x* is the single-character macro name, and *value* is the value assigned to *x*.

Macros and classes can use the same letters for their names. For example, you can have a class `A` and a macro `A`.

The value of a macro is interpolated when the configuration file is read. The macro is referenced as `$`*x*, where *x* is the macro name. Conditionals are specified with the following syntax:

`$?x` *text1* `$|` *text2* `$.`

This interpolates *text1* if macro `x` is set, and *text2* otherwise. The `$.` terminates the conditional clause. The else clause (`$|`) is optional.

Table 23 lists predefined macros. Table 24 lists macros that must be defined for `sendmail` to work.

Table 23 Predefined macros

Macro	Defines
a	Origination date in Internet format. Time is taken from the `Date:` header line or, if there is no `Date:` header line in the incoming message, `$a` is set to the current date.
b	Current date and time in Internet format used in timestamps.
c	Hop count, or number of times a given message is processed by `sendmail`.
d	Origination date in UNIX (`ctime`) format, equivalent to the `$a` macro.
f	Sender (`From:`) address.
g	Sender address relative to the recipient.
h	Recipient host, set from the `$@` part of rule set 0.
i	Message queue ID in the form *ZZAAnnnnn* on a given host. Used in the `Message-Id:` and `Received:` header lines. (Refer to the mqueue(5) man page for information on the message queue ID.)
p	`sendmail`'s process ID (PID). Can be used to create unique message IDs.
r	Protocol used to receive the message.
s	Sender host name.
t	Numeric representation of Greenwich Mean time. Can be used to create unique message IDs.
u	Recipient user, set from `$:` part of right-side rule that resolves address in rule set 0.
v	Version number of `sendmail`. Used in SMTP banner.
w	Canonical or official name of this host as returned by `gethostbyname`.
x	Full name of sender. The sender's full name is determined from header information, passed with the `-F` command line flag to `sendmail`, or taken from the `passwd` database if the message originates locally.
y	The sender's terminal ID (tty).
z	Home directory of the recipient, if local.

Table 24 Required macros

Macro	Defines
e	SMTP banner. The $e macro is the first output when the SMTP server is invoked. The first word of macro e must be the $j macro, which should be your host's canonical name.
j	Canonical or official name of this host, as returned by gethostbyname().
l	Format of UNIX From_ line.
n	Name of sender. MAILER-DAEMON for error messages.
o	Set of operators or delimiters in addresses.
q	Default format of sender address, which appears in the From: header field.

Figure 112 shows example definitions with comments for the required macros shown in Table 24. These macro definitions do not necessarily appear together in the sendmail.cf configuration file.

Figure 112 Macro definition example

```
#canonical official hostname
Dj$w
#name of sender for error messages
DnMAILER-DAEMON
#UNIX header format
DlFrom $g $d
#Delimiter (operator) characters
Do.:%@!^=/[]
#Default sender address format
Dq$g$?x ($x)$.
#SMTP login message
De$j Sendmail (CONVEX) $v/$Z ready at $b
```

O: options

O lines set options in the configuration file to control `sendmail` operation. Options can also be set on the command line with the `-o` flag.

The syntax of an O line is

Ooovalue

where *o* is a single-letter option name, and *value* is a valid value for the option. *value* can be a:

- String
- Integer
- Boolean (`t`, `T`, `f`, `F`); default is TRUE.
- Time interval

Time intervals are set using the abbreviations listed in Table 25.

Table 25 Time interval abbreviations

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

For example, the line

```
OT2d
```

sets the timeout option for queued messages, `T`, to 2 days.

For a complete list of `sendmail` configuration options, refer to the sendmail.cf(5) man page.

P: message precedence

Precedence assigns a rank to a class of messages. The rank is used to determine the priority of queued messages (refer to the "Managing `sendmail`" chapter in *Managing ConvexOS: Operations Guide* for information on the mail queue).

The syntax of a `P` line is

`P`*message_class=num*

where *message_class* is a string that identifies the class of messages, and *num* is a number assigned to the class.

When *message_class* is included as a precedence field in a message header, the precedence of *message_class* is set to *num*. Higher numbers define a higher precedence. Numbers less than zero suppress the delivery of error messages. The default is zero, which allows `sendmail` to return error messages to the sender. Figure 113 shows typical `P` entries.

Figure 113 Precedence definition example

```
# Message Precedence
Pfirst-class=0
Pspecial-delivery=100
Pjunk=-100
```

T: trusted users

`T` lines define users who have special permissions in `sendmail`. Trusted users are permitted to override the sender address using the `-f` flag, which is necessary to return an error message to the sender. Typically, trusted users are:

- root
- daemon
- uucp
- network

The syntax of a `T` line is:

`T`*user1 user2...*

Trusted user definitions can be placed on more than one line.

H: header formats

H lines define the format of message headers. sendmail uses these formats to generate header lines if they did not appear in the input message. Formats are defined for header lines required by RFC 822 (default header lines) and for mailers (conditional header lines). Header definitions conform to standards and are rarely changed.

The syntax for header lines is

H[?*mflags*?]*name: template*

where

mflags optionally specifies mailer flags required to generate the header. If the Flags field of a mailer definition (described in the "M: Mailers" section) contains the *mflag* specified in the header definition, the header line is included in messages delivered by that mailer.

name is the name of the header field, for example, Date: or From:. The header field name always begins with an uppercase letter.

template is the variable part of the header and can contain absolute text, macros, conditionals, tabs, new lines, and white space. Headers that continue on the next line are preceded by a tab. Macros in header definitions are expanded before being inserted into the output message.

For example, the definition

```
HReceived: $?sfrom $s $.by $j ($v/$Z)
id $i; $b
```

has a conditional statement in the template part of the definition that says if the message is to be relayed from another host ($?s), include the sender's host name ($s) in the header preceded by the word from and always include the word by with the official domain name ($j) for the recipient host. Each case is followed by the version numbers of sendmail and the sendmail.cf file ($v/$Z) and a new line consisting of a tab, the message ID ($i), and the current date and time in Internet format ($b).

For example:

```
Received: from hosta.cvx.com by hostb.cvx.com (5.64/1.28)
        id AA24136; Tue, 20 Aug 91 14:26:08 -0500
Received: by hosta.cvx.com (5.64/1.28)
        id AA06330; Tue, 20 Aug 91 14:25:57 -0500
```

R: rewriting rules

Rewriting rules form the core of address parsing in `sendmail`. They:

- Transform addresses into the appropriate form for a given network
- Determine the appropriate mailer for a given recipient

Rewriting rules have three parts, separated by tabs:

- The left-hand side (LHS)
- The right-hand side (RHS)
- Optional comment field

When an address matches the LHS, it is rewritten as specified by the RHS. The same rule is reapplied to the rewritten address until the pattern no longer matches. Then the next rule in the rule set is tried. The process repeats until a rule terminates the rule set with `$@` (refer to the subsection "Right-hand side") or until the end of the rule set is reached. If a rule does not match, the address is not rewritten by that rule.

The syntax for specifying a rule is

R*LHS* <**TAB**> *RHS* <**TAB**> [*comments*]

where *LHS* is the left-hand side (pattern) of the rule, *RHS* is the right-hand side (replacement), and *comments* are ignored. Each field is separated by a tab.

Address scanning

`sendmail` begins the rewriting process by scanning an address and splitting it into tokens. Comments are saved and later reinserted. The `$o` macro (listed in Table 24) defines characters that act as token separators (atoms) and are tokens themselves.

For example, with "`@`" included in the `$o` macro, the address

```
jane@hosta
```

would be scanned as three tokens: "jane," "@," and "hosta."

Unquoted spaces are also considered tokens and are replaced in the rewriting process with the character specified by the B configuration option.

Left-hand side

Once the address is parsed into tokens, the tokens are matched to patterns defined in the LHS of the rule. A token matches a pattern by matching literal text, including delimiter characters, or by matching metasymbols. Case is ignored in literal strings.

Metasymbols are characters with special meaning and are introduced with a dollar sign ($). Table 26 lists the LHS metasymbols and their meanings.

Table 26 LHS token metasymbols

Metasymbol	Meaning
$*	Match zero or more tokens
$+	Match one or more tokens
$-	Match exactly one token
$=*x*	Match any token in class *x*
$~*x*	Match any token not in class *x*

A token that matches a metasymbol on the LHS of a rule is referred to as $*n* on the RHS of the rule, where *n* is a number representing the indefinite token. For example, if the LHS of a rule contains

```
$+@$-.UUCP
```

and it is applied to the address

```
jane@hosta.uucp
```

the matches would be

```
$1 = jane
literal match = @
$2 = hosta
literal match = .UUCP
```

Right-hand side

The RHS of a rewriting rule specifies the replacement text when the LHS is matched. The RHS can contain any of the tokens matched from the LHS, literal text, and metasymbols. Table 27 lists the RHS metasymbols and their meanings.

Table 27 RHS token metasymbols

Metasymbol	Meaning
`$`*n*	Substitute the token or tokens that match the *n*th occurrence of a `$+`, `$-`, `$*`, `$=`, or `$~` on the LHS
`$[`*name*`$]`	Replace host name *name* with canonical name returned by gethostent(). name can be a host name or an Internet address.
`$>`*n*	Calls rule set *n*. This syntax causes the remainder of the line to be substituted as usual and then rewritten by rule set *n*. The output of rule set *n* then becomes the replacement text for this rule.
`$#`*mailer*	Resolve to *mailer*. *mailer* must be defined in the configuration file. `$#` must only be used in rule set 0 because it causes the rule set to terminate immediately and informs `sendmail` that the address has completely resolved to the format {*mailer, host, user*}. If the mailer is local (has the `l` flag defined), `$@`*host* must be omitted.The complete syntax is: `$#`*mailer* `$@`*host* `$:`*user*
`$@`*host*	Specifies the host part of the {*mailer, host, user*} format to which the address resolves.
`$:`*user*	Specifies the *user* part of the {*mailer, host, user*} format to which the address resolves.
`$@`	As a prefix, applies the rule once and terminates the rule set, returning the remainder of the RHS as the value.
`$:`	As a prefix, terminates the current rule, but continues the rule set. Required before a call to another rule set to prevent an infinite loop.

If the name server is running, it is called to resolve the Internet address or name in the token `$[`*name*`$]` and look for an MX (Mail Exchanger) record for the best routing. MX records allow mail to use dynamic routing supported by BIND so that mail can be delivered through an alternate route when the preferred route is down.

Rewriting rule examples

Table 28 shows examples of rewriting rules and their application. For this example, addresses are rewritten using the following macro and class definitions:

- `$D`, the domain name, is acme.com.
- `$w`, official host name, is sys1.acme.com.
- Class `w`, all names the host is known by, is sys1, sys1.acme.com, and myhost.

Table 28 Rewriting rule examples

LHS	RHS	Address in	Address out	Comments
`R$+<@$->`	`$@$1<@$2.$D>`	`jpc<@sys1>`	`jpc<@sys1.acme.com>`	`$D` is this domain.
		`vax1::rac`	`vax1::rac`	No match.
`R$+<@$=w>`	`$:$1<@$w>`	`skw<@sys1>`	`skw<@sys1.acme.com>`	Rewrite everything in class `w` as `$w` (official hostname).
		`dft<@sys2>`	`dft<@sys2>`	No match—not in class `w`.

S: rule sets

Rewriting rules are grouped into rule sets. Certain rule sets have specific semantics (for example, they are called for certain address types in a specific order); other rule sets can be created to support mailer definitions or to support other rule sets as "subroutines."

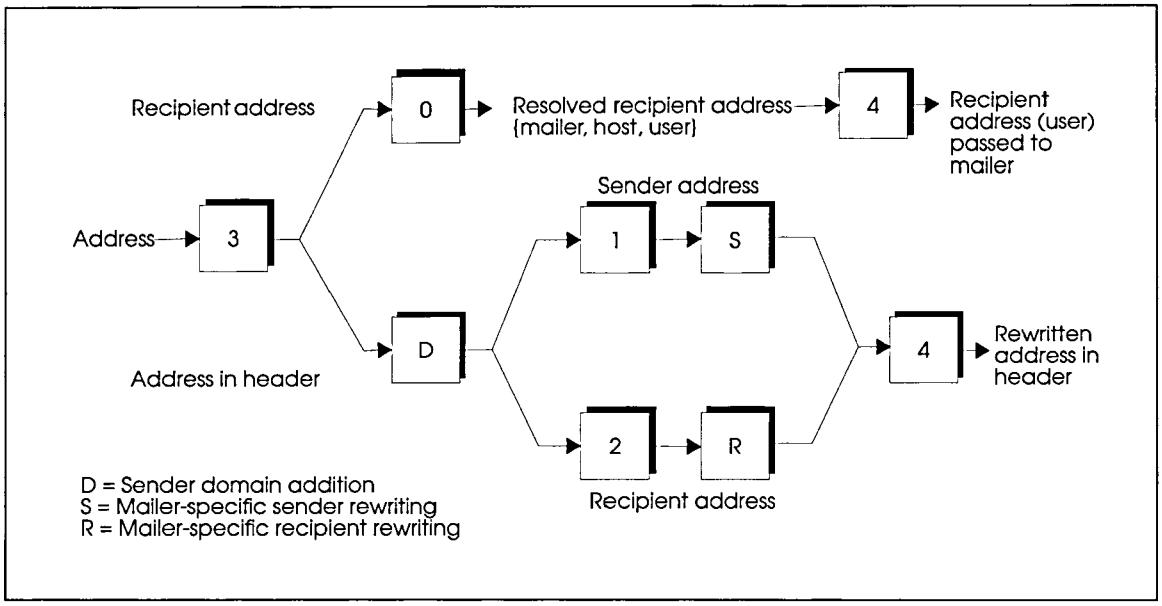
The syntax for specifying the start of a rule set is

`S`*n*

where *n* is the rule set number. All the `R` lines (rules) following `S`*n* are part of rule set *n;* beginning a rule set more than once deletes the previous definition. There cannot be an `R` line before the first `S` line.

Rule sets 0, 1, 2, 3, and 4 are predefined to perform specific functions in a particular order. Additional rule sets can be defined to rewrite mailer-specific headers or to be called as "subroutines" by other rule sets. Figure 114 illustrates how the five predefined rule sets process addresses.

Figure 114 Rewriting rule set flow



The functions of each rule set shown in Figure 114 are described below in the order they appear in that figure, from left to right.

Rule set 3

Rule set 3 is always applied first. This rewriting step transforms addresses into an internal form that is easy to parse by enclosing the host part of the address in angle brackets.

For example,

```
user@host
```

could be rewritten by rule set 3 as

```
user<@host>
```

and

```
host!user
```

as

```
user<@host.uucp>
```

Sender domain addition

The sender domain (D in Figure 114) is added to header addresses that have no domain when the sender's mailer has the `C` flag set and the envelope sender address has an `@` sign. The `C` flag instructs `sendmail` to automatically add the *@domain* clause to any addresses in the header that are local to the sending host, but not local to the receiving host. For example:

```
From: usera@hosta

To userb@hostb, userc
```

would be rewritten as

```
From: usera@hosta

To userb@hostb, userc@hosta
```

Rule set 0

Rule set 0 determines from the recipient address which mailer will be used to deliver a message. Addresses processed by rule set 0 must resolve to {*mailer, host, user*} format. The RHS metasyntax

```
$#mailer$@host$:user
```

specifies the mailer. (Refer to Table 27 for a description of the syntax.)

Once rule set 0 resolves an address, the message is delivered. If the delivery fails, `sendmail` does not try to rewrite the address in an attempt to find another mailer.

You can generate an error message using the "error" mailer. This mailer is used in rule set 0 only. The syntax is

```
$#error[$@status]$:message
```

where *status* is an optional numeric exit status, and *message* describes the error condition. For example:

```
$#error$:Invalid address
```

Rule set 1

Rule set 1 applies protocol-independent transformations to sender addresses. The address rewritten by rule set 1 is then rewritten by a rule set defined for the recipient mailer (designated by an "S" in Figure 114). The number of this rule set is specified in the mailer definition (refer to the section "M: mailers," on page 225).

Rule set 2

Rule set 2 applies protocol-independent transformations to recipient addresses. The address rewritten by rule set 2 is then rewritten by a rule set defined for the recipient mailer (designated by an "R" in Figure 114). The number of this rule set is specified in the mailer definition (refer to the section "M: mailers.").

Rule set 4

Rule set 4, always done last, rewrites addresses from the internal form (done by rule set 3) back into the external form. For example

user<@*host*>

becomes

user@*host*

M: mailers

M lines define the mailers that sendmail uses to deliver mail. sendmail uses mailer definitions to build the command lines that invoke the mailers.

The format of an M line is

M*name*, *field*=*value*, [*field*=*value*,...]

where *name* is the name sendmail uses to refer to the mailer and *field*=*value* pairs define the characteristics of the mailer.

Table 29 lists the mailer fields. Only the first character of *field* is significant, so the first character of each field name is shown in bold face type in Table 29.

Table 29 Mailer fields

Field	Description
Path	Mailer executable path name. If the mailer is accessed over an IPC connection, the string `[IPC]` is used.
Flags	Flags that define how `sendmail` interacts with the mailer or indicate which header fields the mailer requires.
Recipient	Mailer-dependent rule set that rewrites recipient addresses in message header. This rule set is applied after the sending domain is appended and rule sets 1 and 2 are applied, but before the final output rewrite (rule set 4) is applied.
Sender	Mailer-dependent rule set that rewrites sender addresses in message header. This rule set is applied after the sending domain is appended and rule sets 1 and 2 are applied, but before the final output rewrite (rule set 4) is applied.
Eol	End-of-line string. The default is a string containing only a newline. The backslash escapes `\r` (carriage return), `\n` (line feed), `\f` (form feed), and `\b` (backspace) can be used.
Argv	Argument vector to pass. This field can have embedded spaces. If there is no `argv` with a `$u` macro in it, `sendmail` will use SMTP. If the path name for this mailer is `[IPC]`, the `argv` should be `IPC $h` [*port*] where *port* is the optional port number to connect to.
Maxsize	Maximum message length in bytes.

Mailer flags

Table 30 lists the mailer flags used in the F mailer field.

Refer to the section "H: header formats," for information on the interaction between mailer flags and conditional headers.

Table 30 Mailer flags

Flag	Description
A	This is an Internet-compatible mailer; all appropriate modes should be set.
C	Append domain information (*@domain*) to addresses received from this mailer if it is missing. (Refer to the section "Rule Set 3," page 223.)
D	Mailer expects a Date: header line.
E	Ignore lines beginning with "From" in the message by preceding them with a ">" sign.
e	This mailer is expensive to connect to; wait until the next queue run to connect. The c configuration option must also be specified for this flag to take effect. (Refer to the sendmail.cf(5) man page for more information on sendmail configuration options.)
F	Mailer expects a From: header line.
f	This mailer expects a -f flag if this is a network forward operation (the mail gives an error if the executing user is not defined as a trusted user).
h	Preserve uppercase in host names.
I	This mailer uses SMTP and can use special features of the protocol to maximize efficiency. This flag is optional for SMTP transmission.
L	Limit line lengths in messages transmitted via SMTP as specified in RFC 821.
l	This is a local mailer (it performs final delivery).
M	Mailer expects a Message-Id: header line.
m	Multiple recipients can be handled in one transaction.
n	Do not insert a UNIX-style From_ line at the beginning of the message.
P	Mailer expects a Return-Path: header line. RFC 822 specifies that this header line should be used only with mailers that perform final delivery.
p	Use the return-path in the SMTP MAIL FROM: command rather than just the return address. Although this is required in RFC 821, many hosts do not process return paths properly.
r	Same as f, but expects a -r flag (-r is an obsolete form of -f).

Table 30 Mailer flags (continued)

Flag	Description
S	Do not reset the user ID (UID) before calling the mailer. This flag should be specified if this mailer must be called as root. Otherwise, before calling a mailer, `sendmail` resets its UID to the calling user's UID. If running in daemon mode, `sendmail`'s UID is set to the default defined with the `u` option. This flag is suppressed if `sendmail` was started with the `-C` flag or is not running setuid to root.
s	Strip quote characters (backslashes (\) and quotation marks (")) from the address before calling the mailer.
U	Mailer expects UNIX-style `From _` lines with UUCP-style "remote from" on the end.
u	Preserve uppercase in user names.
X	Use the RFC 821 hidden-dot algorithm. Any line beginning with a dot will have an extra dot prepended (to be stripped at the other end). This prevents lines in a message that contain a dot from terminating the message prematurely.
x	Mailer expects a `Full-Name:` header.

Definition examples

Two examples of mailer definitions are given below. Fields are described from left to right.

The following example specifies a mailer that performs local delivery:

```
Mlocal,P=/bin/mail, F=rlsDFMmn, S=10, R=20,
A=mail -d $u
```

The name of this mailer is `local`. Its path name is `/bin/mail`. The `Flags` field specifies that this mailer expects a `-r` flag, is a local mailer, and expects quote characters to be stripped off. The headers `Date:`, `From:`, and `Message-Id:` are expected. This mailer can deliver to multiple users at once. The UNIX-style `From_` header should not be inserted on the front of messages delivered by this mailer.

Rule set 10 is applied to sender addresses and rule set 20 is applied to recipient addresses. The argument used when invoking this mailer is `mail -d` followed by the recipient names (`$u`).

The following example specifies a mailer that performs Ethernet delivery:

```
Msmtp,P=[IPC], F=mDFMueXLC, S=17, R=27,
A=IPC $h, E=\r\n
```

This mailer is called `smtp`. It uses an IPC connection to deliver messages on the default port, 25. It can handle multiple users at once; expects the `Date:`, `From:`, and `Message-Id:` headers; and expects uppercase user names to be preserved. It is identified as an expensive mailer and uses the hidden-dot algorithm. Line lengths are limited as specified in RFC 821, and domain information in addresses received from this mailer is added if missing.

Rule set 17 is applied to sender addresses and rule set 27 is applied to recipient addresses. Because this mailer uses an IPC connection, the `Argv` field contains the word IPC followed by the name of the recipient host (`$h`). This mailer expects carriage return or line feed end-of-line characters.

Modifying the `sendmail` configuration file

This section provides step-by-step instructions for modifying the sendmail.cf file provided with ConvexOS.

Prerequisites

Before you modify the sendmail.cf file, you need to know how you want mail to be routed. Some approaches are:

- Each system on the local network can send mail to any other system on any network.
- Mail hosts are set up for UUCP, Internet, BITNET, and so on. The internal hosts send mail outside the local domain via these mail host relays.
- Some hosts on the local network use relays; other hosts can send mail to any network directly.

The size of your site is a factor in determining which approach to take. A large site can designate one system as a gateway mail host to which other hosts at the site forward mail they cannot deliver. The gateway mail host must be able to resolve all addresses; other hosts at the site need only the minimum information required to forward mail to the gateway. This approach eases maintenance of configuration files. Smaller sites can have complete information on each host.

You should also know:

- Your system's UUCP name. This is usually the same as the name returned by the `hostname` command.
- The UUCP sites you can connect to directly.
- The name of your UUCP relay host, if you have one.
- If your system accesses the Internet. If it does, your host address should be registered with the Network Information Center (NIC). Refer to *Internet Services System Manager's Guide*, the chapter "Setting Up the Host Name Database."
- If you want to use the NIS aliases map instead of /usr/lib/aliases if you are running NIS.

If you have more than one host at your site or you have a site that uses only UUCP, you can automate modification of the configuration files using `m4` conditional substitutions. Otherwise, modify the configuration file directly using the procedure provided in this section. If you choose to use the `m4` method, instructions are provided in the file /usr/lib/conf/sendmail/convex/README.

Procedure

The sendmail configuration file path name is /usr/lib/sendmail.cf.V*x.x*, where *x.x* is the version of ConvexOS; for example, sendmail.cf.V10.0.

The word "SETUP:" in the sendmail.cf file precedes a line or section that you may need to change by typing in information and adding or deleting a "#" sign. Anything following a "#" sign is a comment and is not interpreted by `sendmail`.

Each "SETUP:" comment in the configuration file has a corresponding step in this procedure.

The word "CONFIG:" precedes lines or sections in the file that define options. CONFIG: lines are not included in this procedure because `sendmail` will work with the options as they are set in the supplied sendmail.cf file.

To modify the sendmail.cf file, follow these steps:

Step 1 Change to your work directory:

```
# cd
```

Step 2 Copy the supplied sendmail.cf file:

```
# cp /usr/lib/sendmail.cf.V10.0 sm.cf
```

Step 3 Use an editor to open the sm.cf file and search for the string SETUP:.

The first SETUP: string is shown in Figure 115.

Figure 115 Define the w class

```
# SETUP: Define the w class.
#         : If F is used, the file containing our internet aliases.
#         : If C is used, the string containing our internet aliases
#         : You should not have to uncomment this unless you wish to have
#         : aliases that are not defined with the hostname entry!
#Fw/usr/lib/sendmail.cw
#Cwyourhost yourhost.subdom.dom
```

Step 4 If you have host name aliases that are not defined in any host name database (/etc/hosts, hosts.map, or named.hosts), type all the names your host is known by in a separate file or in the configuration file on the line preceded by `#Cw`. If you type the host names in the separate file, remove the "#" sign from the `F` line and type the full path name of the file as shown in Figure 115. If you type the host names in the configuration file, remove the "#" from the `C` line. If you do not have host name aliases that are not in one of the host name databases, go to the next step.

Step 5 Search for the next SETUP:, "Define the U macro."

Step 6 If you have local UUCP connections, enter your canonical UUCP host name to define the `U` macro, as shown in Figure 116. The UUCP host name is usually the same as the name returned by the `hostname` command. If you do not directly connect to UUCP hosts, go to Step 11.

Figure 116 UUCP name

```
# SETUP: Define the U macro.
# : Our UUCP hostname (usually the same as /bin/hostname).
#DU YOUR_UUCP_NAME_GOES_HERE
```

Step 7 Search for the next SETUP:, "Define the U class."

Step 8 If you defined a UUCP name, define the `U` class by listing any UUCP aliases for your machine, as shown in Figure 117.

Figure 117 UUCP aliases

```
# SETUP: Define the U class.
# : Our UUCP hostname aliases.
#CU YOUR_UUCP_ALIASES_GO_HERE
```

Step 9 Search for the next SETUP:, "Define the V class."

Step 10 Define the `V` class as the list of local UUCP hosts (neighbors) that you can connect to directly over UUCP, as shown in Figure 118. You can list these hosts in a separate file or in the configuration file.

When using a separate file, remember that you can obtain the host names from a file such as /usr/lib/uucp/L.sys by specifying a `scanf` format. Remove the "#" sign from the appropriate line.

To define the V class as a file, enter the path name of the file containing the host names and remove the "#" sign from the F line. To define the V class in the configuration file, enter the host names on a single C line or on multiple C lines as shown in Figure 118. Remove the "#" sign from the appropriate line or lines.

Figure 118 UUCP neighbors

```
# SETUP: Define the V class.
# : List of local UUCP connections (neighbors) you can connect to
# : directly over UUCP.
#FV/usr/lib/sendmail.V.uucp
#CVuucphost1 uucphost2
#CV UUCP_HOST1_WE_CONNECT_TO_DIRECTLY
#CV UUCP_HOST2_WE_CONNECT_TO_DIRECTLY
```

Step 11 Search for the next SETUP:, "Define the D macro."

Step 12 The D macro defines your domain name. For example, if your official hostname is host.acme.com, the domain name is acme.com. This definition is required. To find your domain name, look in the files /etc/named.boot or /etc/resolv.conf.

Enter your Internet domain name to define the D macro, as shown in Figure 119.

Figure 119 Domain name

```
# SETUP: Define the D macro.
# : The local domain name (if you use domain names even without the
# : nameserver set this to the domain.
DD YOUR_DOMAIN_GOES_HERE
```

Step 13 Search for the next SETUP:, "Define the j macro."

Step 14 The j macro, official name of the local host, is set in the supplied configuration file to $w, which also is the official name of the local host (see Figure 120). You can define the j macro as $w.$D if your official host name as defined by $w does not already include the domain.

If you want to use $w.$D, add a "#" sign to the Dj$w line and remove the "#" from the Dj$w.D line.

Figure 120 Official host name

```
# SETUP: Define the j macro.
# : Our canonical (official) hostname.
# : Set by gethostbyname(2).
# : Change this (by adding the $D macro), only if all of the
# : following are true:
# : 1. you are not using the nameserver
# : 2. your hostname is the short version (not the FQDN)
# : 3. you wish to have a FQDN
Dj$w
#Dj$w.$D
```

Step 15 Search for the next SETUP:, "Define the A macro."

Step 16 The `A` macro defines your Internet relay host for the `From:` header line. This macro is required. The Internet relay host hides machines in your domain and produces addresses of the form

user@relay.domain

although delivery is still performed by your system. The `A` macro is defined in the supplied configuration file to be your official host name (`$w`). You can also designate another host or your domain name (`$D`) as the Internet relay host.

To define the `A` macro as another host, enter the name of the Internet relay host as shown in Figure 121. Remove the "`#`" sign from the appropriate line and insert a "`#`" sign in front of the line "`DA $w.`"

To use `$D`, remove the "`#`" sign from the appropriate line and insert a "`#`" sign in front of the line "`DA $w.`"

Figure 121 Internet relay

```
# SETUP: Define the A macro.
# : Internet relay host for sender address rewriting
# : Machines in our domain will be "hidden" behind this relay machine
# : with the notation user@relay.dom, although SMTP delivery will
# : still be performed by the this sending machine.
# : Must be set! May want to use the domain name ($D) or official
# : hostname ($w) macro.
#DA $D
DA $w
#DA YOUR_INTERNET_RELAY_HOSTNAME_GOES_HERE
```

Step 17 Search for the next SETUP:, "Define the G macro."

Step 18 The G macro defines the mail host designated to handle all mail in and out of your domain. Internal host names may not appear in addresses if a gateway is defined.

If you want to have an Internet gateway, enter the name of the host to define the G macro as shown in Figure 122 and remove the "#" sign.

Figure 122 Internet gateway

```
# SETUP: Define the G macro.
#         : The name of the host you wish to have all internal hosts
#         : forward mail to for external (outside our domain ($D))
                  address
#         : delivery
#DG YOUR_INTERNET_GATEWAY_GOES_HERE
```

Step 19 Search for the next SETUP:, "Define the R macro."

Step 20 The R macro defines your UUCP relay host. If defined, this is the machine to which nonlocal UUCP mail is forwarded. That is, any address that ends in .UUCP or is of the form *host*!*user* that is not listed in the V class is forwarded to the UUCP relay host via SMTP.

If you have a UUCP relay, enter the Fully Qualified Domain Name (FQDN) of the host to define the R macro as shown in Figure 123 and remove the "#" sign.

Figure 123 UUCP relay

```
# SETUP: Define the R macro (the UUCP relay host).
# : Send UUCP addresses (ending with .UUCP - user@host.UUCP or
                  host!user)
# : to this UUCP relay host (a SMTP host) for processing.
# : Should probably set to a FQDN of the UUCP relay host.
#DR YOUR_UUCP_RELAY_GOES_HERE
```

Step 21 Search for the next SETUP:, "Define the B macro."

Step 22 The B macro defines your BITNET relay host. If defined, this is the machine to which BITNET mail (mail addressed to *user*@*host*.BITNET) is forwarded via SMTP. If not defined, mail addresses of the form *user*@*host*.BITNET will bounce.

If you have a BITNET relay, enter the FQDN of the host to define the B macro as shown in Figure 124 and remove the "#" sign.

Figure 124 BITNET relay

```
# SETUP: Define the B macro (the BITNET relay host).
# : Send BITNET addresses (ending with .BITNET - user@host.BITNET) to
# : this BITNET relay host (a SMTP host) for processing.
# : Should probably set to a FQDN of the BITNET relay host.
#DB YOUR_BITNET_RELAY_HOSTNAME_GOES_HERE
```

Step 23 Search for the next SETUP:, "Define the E macro."

Step 24 The E macro defines your DECNET relay host. If defined, this is the machine to which DECNET mail (mail addressed to *user@host*.DNET) is forwarded via SMTP. If not defined, mail addresses of the form *user@host*.DNET will bounce.

If you have a DECNET relay, enter the FQDN of the host to define the E macro as shown in Figure 125 and remove the "#" sign.

Figure 125 DECNET relay

```
# SETUP: Define the E macro (the DECNET relay host).
# : Send DECNET addresses (ending with .DNET - user@host.DNET) to
# : this DECNET relay host (a SMTP host) for processing.
#DE YOUR_DECNET_RELAY_HOSTNAME_GOES_HERE
```

Step 25 Search for the next SETUP:, "Define the E class."

Step 26 Define the E class as the list of local DECNET connections (neighbors) that you can connect to directly over COVUEnet or the DECNET relay host. Define this class only if you defined a DECNET relay host or have a COVUEnet mailer defined in the sendmail.cf file. You can list these hosts in a separate file or in the configuration file.

To define the E class as a file, enter the path name of the file containing the host names and remove the "#" sign from the F line. To define the E class in the configuration file, enter the host names on the C line and remove the "#" sign, as shown in Figure 126.

Figure 126 DECNET neighbors

```
# SETUP: Define the E class.
# : List of local DECNET connections (neighbors) you can connect to
# : directly over COVUEnet or the DECNET relay.
# : This class defines known DECNET hosts and requires that either
# : COVUENET (mailer) is defined or a DECNET_RELAY is defined.
#FE /usr/lib/sendmail.E.decnet-covue
#CE decnet/covue_host1 decnet/covue_host2
```

Step 27 Search for the next SETUP:, "Define the I class."

Step 28 The `I` class defines "fake" domains. These are domains that are not recognized in the Internet domain.

If you did not define relay hosts for UUCP, BITNET, or DECNET and do not want to send mail to any of these fake domains, insert a "#" sign at the beginning of the line shown in Figure 127.

Figure 127 Fake domains

```
# SETUP: Define the I class.
# : Resolve fake (top level) domains by forwarding to designated
# : relay hosts for each of the defined domains.
# : The domains are not known by the (NIC Registered)
# : Domain Name System (DNS).
CIUUCP BITNET DNET
```

Step 29 Search for the next SETUP:, "Use the NIS (YP) mail.alias file instead of normal alias file."

Step 30 If you are running Network Information Services (NIS), you can use the NIS mail.aliases file or /usr/lib/aliases. The supplied configuration file uses the mail.aliases file. If NIS is not running, /usr/lib/aliases is used.

If you are running NIS and want to use /usr/lib/aliases, add a "#" sign before the `p` option shown in Figure 128.

Figure 128 Use NIS mail.alias file

```
# SETUP: Use the NIS (YP) mail.alias file instead of normal alias file
# : If yp is not running then defaults to /usr/lib/aliases
Op
```

Step 31 Search for the next SETUP:, "Use the mail gateway host."

Step 32 If you have a mail gateway defined (the G macro is defined) and you want non-local domain mail to go out through this gateway, remove the "#" from the rule shown in Figure 129. If you are not using the name server, the recipient host must be in the gateway host's NIS hosts map or /etc/hosts file on the gateway system.

Figure 129 Send non-local domain mail via the gateway

```
# SETUP: Use the mail gateway host (define G macro) for external
                domains.
# : If you want to send non-local domain mail via your mail
# : gateway host (mailhost), uncomment (#) the following rule.
# : If this host is the mail gateway and you are not using the
# : nameserver, the recipient hosts must be in the NIS hosts map
# : or in /etc/hosts file on the gateway system.
#R$+<@$+.$~D>$*$#smtp$@$G$:$1<@$2.$3>$4usr@host.rdom via gtwy
```

Step 33 Test `sendmail` with the new configuration file using any of the methods explained in the next section, "Testing the configuration."

Step 34 Copy the old `sendmail` configuration file to another file (you must be root):

```
# cp /usr/lib/sendmail.cf\
/usr/lib/sendmail.cf.MMDDYY
```

MMDDYY is the date (for example, `091291`).

Step 35 Copy the modified `sendmail` configuration file sm.cf to sendmail.cf:

```
# cp sm.cf /usr/lib/sendmail.cf
```

Step 36 Freeze the configuration file by issuing the following command as root:

```
# /usr/lib/sendmail -bz
```

Step 37 Kill the `sendmail` daemon by finding the PID of the `sendmail` process called `accepting connections` and kill the process using the `kill` command as shown in Figure 130.

Figure 130 Killing the `sendmail` daemon

```
# ps aux | grep accepting
root 218 0.0 0.0 456 4 ? I 0:03 -accepting connections
# kill 218
```

Do not use `kill -9` to kill `sendmail`; this could corrupt the aliases database. Issue the `kill` command without the first argument to send signal 15 to the process.

Step 38 Restart the `sendmail` daemon:

```
# /usr/lib/sendmail -bd -q30m
```

Testing the configuration

sendmail provides extensive testing and debugging facilities. The following sections describe the sendmail debug flag and four methods for testing the configuration.

Debugging

The sendmail debugging facility allows you to specify the type and amount of debugging information to display. When sendmail is invoked with the -d flag, debugging output is directed to the screen.

The syntax for the -d flag is

```
sendmail -dflag.level
```

flag is a number from 0 to 99 that specifies what debug flag to turn on. Range 21 is the most commonly used and performs rule set debugging. Other flags print out address information or values of arguments to various sendmail program functions.

level is a number from 0 to 99 that specifies the amount of information, or level of detail, to display. A level higher than 9 is usually used only to debug sendmail code.

Testing address rewriting

sendmail can be run in test mode that shows how addresses are rewritten by specified rule sets. The command to run sendmail in address test mode is

```
sendmail -bt -Cconfig_file
```

where *config_file* is the name of the sendmail configuration file to be tested. In this mode, sendmail expects lines of the form

rulesets addresses

where *rulesets* is either a single rule set number or a list of rule sets separated by commas but no spaces, and *addresses* is a single address or a list of addresses separated by commas.

After sendmail breaks the address into tokens, it always applies rule set 3 first, followed by the specified rule sets. To see what mailer an address resolves to, use rule set 0. Rule sets 1, S (mailer-specific sender address rewriting), and 4 test sender addresses, and rule sets 2, R (mailer-specific recipient address rewriting), and 4 test recipient addresses. You can test a single rule by creating a rule set containing only that rule.

To exit test mode, type **CTRL-d**.

Figure 131 shows an example of testing an address to see if it resolves to the desired format (*mailer, host, user_address*).

Figure 131 Testing address rewriting

```
% /usr/lib/sendmail -bt
ADDRESS TEST MODE
Enter <ruleset> <address>
> 0, user@host
rewrite: ruleset 3 input: "user" "@" "host"
rewrite: ruleset 8 input: "user" "@" "host"
rewrite: ruleset 8 returns: "user" "@" "host"
rewrite: ruleset 6 input: "user" "<" "@" "host" ">"
rewrite: ruleset 6 returns: "user" "<" "@" "host" ">"
rewrite: ruleset 3 returns: "user" "<" "@" "host" ">"
rewrite: ruleset 0 input: "user" "<" "@" "host" ">"
rewrite: ruleset 3 input: "user" "@" "host"
rewrite: ruleset 8 input: "user" "@" "host"
rewrite: ruleset 8 returns: "user" "@" "host"
rewrite: ruleset 6 input: "user" "<" "@" "host" ">"
rewrite: ruleset 6 returns: "user" "<" "@" "host" ">"
rewrite: ruleset 3 returns: "user" "<" "@" "host" ">"
rewrite: ruleset 6 input: "user" "<" "@" "host" ">"
rewrite: ruleset 6 returns: "user" "<" "@" "host" ">"
rewrite: ruleset 0 returns: "^V" "smtp" "^W" "host" "^X" "user" "<" "@" "host"
">"
>
```

The symbols (`^V`, `^W`, and `^X`) in the last line of output in Figure 131 are used by `sendmail` to represent tokens internally.

For address rewriting testing to be useful, you must understand which rules are applied to which addresses and when. If you need to do extensive testing, it is helpful to create a table that lists the mailers in your configuration with the numbers of the rule sets that decide the destination, and recipient and sender addresses for that mailer.

Testing recipient address resolution

To verify that an address is deliverable, invoke `sendmail` with the `-bv` flag. The syntax is

```
/usr/lib/sendmail -bv -v -L10 address
```

where

`-bv` instructs `sendmail` to verify addresses only and does not send or collect messages.

`-v` (verbose flag) reports alias expansion and duplicate suppression.

`-L10` sets the log level to 10, which causes `sendmail -bv` to report the mailer and host for the given recipient address.

For example:

```
# /usr/lib/sendmail -bv -v -oL10 jane
jane... aliased to jane@hosta
jane... deliverable
```

This mode is often combined with `-C`*config_file* to specify an alternate configuration file.

MX (Mail Exchanger) hosts for addresses that resolve to `[IPC]` are not listed in verify mode because MX records are not collected until delivery is attempted.

Verifying mail delivery

`mail` can be run with the `-v` flag to show details of message delivery. This can be used to check sender and recipient host names and verify that the machines can communicate. Figure 132 shows the output of `mail -v` when sending mail to a remote host.

Figure 132 Verifying mail delivery

```
/usr/ucb/mail -v biff@grr.ruf.woof.edu
Subject: SMTP testing
test
.
Cc:
biff@grr.ruf.woof.edu... Connecting to
hosta.cvx.com (smtp)...
220 hosta.cvx.com Sendmail 5.61/1.35 ready at Fri, 20 Sep 91 14:00:19
                    -0500
>>> HELO hostb.cvx.com
250 hosta.cvx.com Hello hostb.cvx.com, pleased to meet you
>>> MAIL From:<jpc@hostb.cvx.com>
250 <jpc@hostb.cvx.com>... Sender ok
>>> RCPT To:<biff@grr.ruf.woof.edu>
250 <biff@grr.ruf.woof.edu>... Recipient ok
>>> DATA
354 Enter mail, end with "." on a line by itself
>>> .
250 Ok
>>> QUIT
221 hosta.cvx.com closing connection
biff@grr.ruf.woof.edu... Sent
```

Testing direct mailing

You can talk to the `sendmail` daemon and other SMTP servers directly with the command

`telnet` *host* `25`

where *host* is the name of the host you are connecting to, and `25` is the port used by `sendmail` to listen for connect requests from other `sendmail` processes across the Internet.

You can use this command to determine if an SMTP server is running on a particular host. If not, the message "`Connection refused`" is returned.

Once connected, four-character SMTP commands are used to communicate with the remote server. SMTP is not case sensitive. The server responds to each command with a three-digit code (followed by explanatory text) that indicates success or failure. `sendmail` implements a subset of the complete SMTP command set. Refer to RFC 821 for more information.

The SMTP VRFY command can be used to verify if the server can route to a particular address. Figure 133 shows using the SMTP VRFY command.

Figure 133 Testing direct mailing

```
% telnet hosta 25
Trying 1.1.1.1...
Connected to hosta.cvx.com.
Escape character is '^]'.
220 hosta.cvx.com Sendmail 5.64/1.28 ready at Mon, 16 Sep 91 18:00:46
                    -0500
vrfy jane
250 Jane User <jane>
vrfy opus
550 opus... User unknown
```

Setting up the notesfile system

11

The notesfile system is a computer bulletin-board system. Notesfiles in the system can be set up as either local notesfile systems or as networked notesfile systems. You can use local notesfiles to:

- Coordinate group discussions
- Provide a resource for problem reporting
- Organize collections of data such as mail messages for individual users

Each notesfile discusses a single topic of common interest to a group of people. Anyone with access to the notesfile can post an original note, called a base note, to the notesfile. Typically, each notesfile contains a number of base notes.

Each base note can have any number of responses, which are comments or related questions concerning the base note. Thus, a notesfile contains an ordered list of base notes and their responses.

The depth of discussion within a notesfile is ideally held constant. If one group needs high-level overview information on a topic, and another group needs in-depth detail of the same topic, the discussions should be separated into two different notesfiles.

This chapter discusses how to set up the notesfile system at your site.

Control of notesfiles

The notesfile system is installed by a user who is known as the owner of the notesfiles. The owner can create, delete, rename, and initiate networking of notesfiles. The owner rarely manages the daily aspects of a notesfile, although the owner has director, read, write, and response privileges to all notesfiles for handling emergencies and failures.

Each notesfile is assigned a director or set of directors who have special privileges for managing the notesfile. A director can:

- Change access permissions for a notesfile
- Write or edit the policy note
- Change the notesfile title and director message
- Open or close the notesfile
- Allow the notesfile to be networked
- Permit or restrict anonymous notes
- Compress the notesfile
- Change the notesfile's archival parameters
- Delete notes and responses
- Change the director message on any note or response

A notesfile is created with a default access list. Group and system are given read and write access to the files. You can set other default access rights using the access-template file. This file contains lines of access rights that are applied to any newly created notesfiles.

Creating notesfiles

To set up the notesfile system, perform the following steps:

Step 1 Log in as the superuser.

Step 2 Change to directory /usr/spool/notes/.utilities.

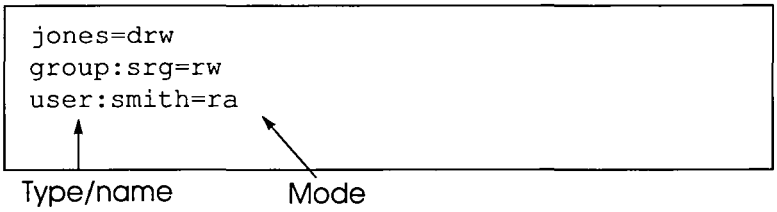
```
# cd /usr/spool/notes/.utilities
```

Step 3 Change user ID to user notes. Enter:

```
# su notes
```

Step 4 If you want to establish default access-right parameters before creating a notesfile, edit a file called access-template in /usr/spool/notes/.utilities. Add any access-right defaults you want applied when a notesfile is created. A sample access-template file is shown in Figure 134.

Figure 134 Sample access-template file



Each line in this file represents an access-right parameter. The format for this file is

type:name=mode

where

type is the user class this parameter affects. This can be `user`, `group`, or `system`. If type is left blank, `user` is assumed.

name is the user or group name this parameter affects.

mode is the permissions granted to the specified user class. This can be any one or combination of the following:

- `d` Provides director privileges to manage the notesfile.
- `r` Provides read privileges to the notesfile.
- `w` Provides write privileges to the notesfile.
- `n` Denies all privileges. Cannot access the notesfile.
- `a` Provides privileges to respond to notes (answer) but not write them.

This file grants user jones director, read, and write privileges, group srg read and write privileges, and user smith read and answer privileges.

Step 5 Create any desired notesfiles using the `mknf` command. Use the format

```
mknf [options...] topic [topic ...]
```

where

options specifies permissions to the notesfile. This can be:

- `a` Permits anonymous notes
- `o` Creates the notesfile open (new notes can be posted to the notesfile)
- `n` Allows the notesfile to be networked

topic is the name of the notesfile. To specify multiple notesfiles with one command, separate each notesfile name with a blank space.

The created notesfiles have a default status of closed, nonnetworked, and no anonymous notes permitted. See the mknf(8) man page for more details on this command.

The notesfile system provides for intersystem notesfiles in a networked system. A notesfile with the same name must exist on each system in the network that wishes to share the notesfile information. The contents are kept in synchronization through exchanges over a network. Notesfiles to be shared must have their network status enabled. See the nfaccess(8) man page for more details.

Step 6 Establish one or more directors for the notesfile using the `nfaccess` command. For example, the following command gives user smith director, read, and write privileges for the notesfile called project_notes.

```
# nfaccess smith=drw project_notes
```

Step 7 If you share notesfiles over a networked system, the notesfile owner must occasionally update remote notesfiles with new notes and receive new remote notes using the `nfxmit` command. The `nfxmit` command gathers the new notes and responses in specified notesfiles and sends them to a specified system.

Use the following format for the `nfxmit` command

`nfxmit -d`*sitename topic* [*topic ...*]

where

sitename is the name of the remote site to receive the new notes. The remote site should have a notesfile matching those specified by the topic parameter.

topic is a single notesfile name or list of names to be updated.

See the nfxmit(8) man page for more information on optional parameters available with this command.

Step 8 Transfer of notesfiles over a networked system can be done automatically by `cron` if you set up the `nfxmit` command in the crontab file. `cron` executes commands at specified dates and times according to the instructions found in the /.crontab file.

Figure 135 illustrates a sample /.crontab file that automates transfer of the notesfile named project_notes to the machine named convex every day of the month, every hour.

Figure 135 Sample /.crontab file

```
59 * * * * nfxmit -dconvex project_notes
```

Each line of the crontab file represents one activity; each field in this line is 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 can be any number between 0 and 59.

hour can be any number between 0 and 23.

date can be any number between 1 and 31.

month can be any number between 1 and 12.

day can be any number between 1 and 7, where 1 equals Monday, 2 equals Tuesday, 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 one 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.

See the crontab(5) man page for more details on specifying commands.

Step 9 Using an editor, add the new notesfile names to the /usr/spool/notes/.utilities/avail.notes file. This file contains a list of public notesfiles. The content and format of this file are at the discretion of the notesfile system owner.

Setting up log files

12

Log files collect activity that can be helpful for tracking purposes. In addition to the accounting logging facilities discussed in Chapter 8, "Setting up the accounting system," three other logging facilities are available with ConvexOS:

- Failed file access logging
- System message logging
- System availability logging

As shipped, none of these facilities are active. This chapter describes each of these logging facilities, their uses, and how to activate them.

Failed file-access logging

For additional system security, ConvexOS provides a facility for logging file-access attempts that fail because of insufficient file-access permissions. With this facility, you can track unauthorized attempts to access protected files or directories. Each file-access attempt that fails due to insufficient permissions is logged to the /usr/adm/failure_log file along with enough information for you to determine who attempted the access and what command was used, the date and time the attempt was made, and the file involved.

The following system calls generate log messages when they fail due to insufficient permissions:

- `access`
- `acct`
- `bind`
- `chdir`
- `chmod`
- `chown`
- `connect`
- `creat`
- `execve`
- `exportfs`
- `faillog`
- `lionk`
- `lstat`
- `mkdir`
- `mknod`
- `open`
- `mount`
- `quotacl`
- `relink`
- `rename`
- `stat`
- `statfs`
- `swapon`
- `symlink`
- `truncate`
- `unlink`
- `unmount`
- `utime`

An attempt to generate a core file also generates a log entry.

Initiating failed file-access logging

Perform the following steps to initiate failed file-access logging. Enabling this utility increases system overhead and degrades system performance.

Step 1 Log in as the superuser.

Step 2 The file that will record the failures must exist before enabling logging. Check to be sure a file called failure_log exists in the /usr/adm directory. If not, create one using the `touch` command. Enter:

```
# touch /usr/adm/failure_log
```

Step 3 Enable logging using the `faillogon` command. You must name the log file where failed attempts are recorded. This file is typically called /usr/adm/failure_log. Enter:

```
# faillogon /usr/adm/failure_log
```

Step 4 If you want failed file-access logging to start automatically each time the system is booted, place the following line in the rc.local file using an editor. Enter:

```
# faillogon /usr/adm/failure_log
```

Step 5 Set the boot-time tunable parameters that control when logging is suspended and resumed. These are the `logsuspend` and `logresume` parameters.

The `logsuspend` parameter specifies the percent of free disk space on the file system that must be available for logging to continue. When the free space falls below the number specified in the `logsuspend` parameter, logging is suspended and a message is written to the system console and to the error log.

The `logresume` parameter specifies the percent of free space on the log file system necessary for logging to resume after it has been suspended.

Refer to Chapter 15, "Customizing kernel boot-time parameters," on page 287, for details on how to set tunable parameters.

Step 6 Create a shell script that will automatically execute the `faillogpr` utility, rename old log files, and create a new log file on a daily basis.

For example, Figure 136 illustrates a sample script. You must run this script as the superuser.

Figure 136 Sample script for maintaining failure_log files

```
tmp="/usr/adm/fl.$$"
set 'date'; permanent="/usr/adm/faillog/faillog.$6.$2.$3.$4"
mv /usr/adm/failure_log $tmp
touch /usr/adm/failure_log
chmod 600 /usr/adm/failure_log
/etc/faillogon /usr/adm/failure_log
/usr/adm/faillogpr $tmp > $permanent
rm $tmp
```

The first line establishes the naming convention for the variable called *$tmp*. This is /usr/adm/fl.*xxxx*, where *xxxx* is the process ID (PID) of the shell. The contents of the failure_log file will be moved here. This file must exist on the same file system as the /usr/adm/failure_log file. This way, the `mv` command renames rather than copies.

The second line sets a timestamp on *$permanent*. *$permanent* is assigned the file name where the permanent, formatted copy of the log is kept.

The third line moves the contents of the failure_log file to *$tmp* while logging continues. *$tmp* becomes the open log file and continues to receive entries until a new log file is designated.

The fourth line creates a new log file.

The fifth line changes the access mode of the log file.

The sixth line switches logging from the *$tmp* file to the new /usr/adm/failure_log file.

The seventh line formats the *$tmp* file, resolves the device and inode numbers into a path name and sends the output to the *$permanent* file. Keep this file for as long as it is needed.

The last line removes the unformatted log, *$tmp*, because it is no longer needed.

Step 7 Place an entry in the /usr/lib/.crontab file to run the shell script created in Step 6. `cron` executes commands at specified dates and times according to the instructions found in the /usr/lib/.crontab file.

Figure 137 illustrates a sample /usr/lib/.crontab file that automates running the shell program named /usr/log_clean every day at 6:30 a.m.

Figure 137 Sample /usr/lib/.crontab file

```
30 6 * * * /usr/log_clean
```

Each line of the crontab file represents one activity; each field in this line is 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	can be any number between 0 and 59.
hour	can be any number between 0 and 23.
date	can be any number between 1 and 31.
month	can be any number between 1 and 12.
day	can be any number between 1 and 7, where 1 equals Monday, 2 equals Tuesday, 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 one 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.

See the crontab(5) man page for more details on specifying commands.

Printing log information

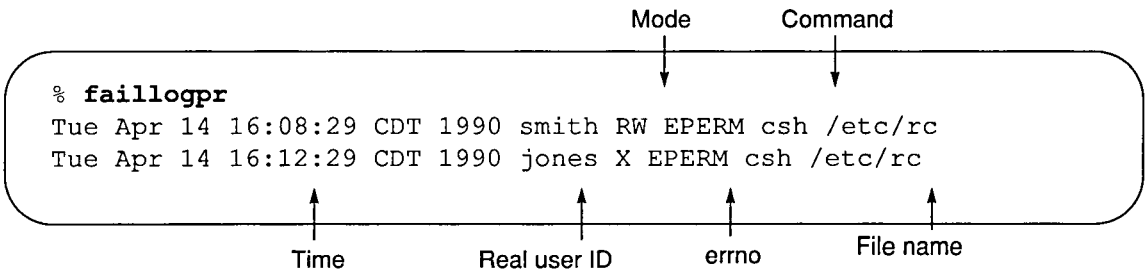
The /usr/adm/failure_log file actually stores the major and minor device numbers and the inode number for the file rather than the full path name. The `faillogpr` program translates the information stored in the failure_log file, including expansion of the file information into a full path name, and displays all of this in a readable report format.

To print the formatted log file to the terminal screen, enter the following command as the superuser:

```
# faillogpr /usr/adm/failure_log
```

Figure 138 illustrates sample output from this command.

Figure 138 Output from `faillogpr` command



The `faillogpr` program displays the following information:

Time	Date and time of attempted access.
RUID	Real user ID of user attempting access.
EUID	Effective user ID of user attempting access, if different from the real user ID.
Mode	File-access mode (read, write, execute) of the file.
errno	Mnemonic of error.
Command	Command that generated error.
File name	File user attempted to access. File names generated by `faillogpr` are the correct file names at the time `faillogpr` is run, not when the failed access occurred. If the file associated with the inode listed in the failure_log file is removed, the inode may be allocated to a new file before the `faillogpr` command is used. If this happens, the file name listed is incorrect.

Stopping file-access logging

To disable logging, use the `faillogon` command without an argument. For example, enter:

```
# faillogon
```

Configuring system message logging

ConvexOS provides a facility that logs user-specified messages to user-specified files. This can be the same file for all message types or any number of files for different message types. You can have messages sent to one or more of the following places:

- The terminal
- A file
- A user
- Other systems

Entries in the /etc/syslog.conf file control what types of messages are logged and where they are sent. Use the following procedure to set up the syslog.conf file according to your site's needs.

Step 1 Log in as the superuser.

Step 2 Modify the syslog.conf file. Each line in the syslog.conf file represents a message group. The format for this file is:

facility.*level* *send_message_here*

where

facility	is the part of the system that generates the message. This can be:
`kern`	Messages generated by the kernel.
`user`	Messages generated by the user.
`mail`	Messages generated by the mail system.
`daemon`	Messages generated by the system daemons.
`auth`	Messages generated by the authorization system, which consists of `login`, `su`, or `getty`.
`lpr`	Messages generated by the line printer spooling system.
`syslog`	Messages generated by the syslog utility.
`news`	Messages generated by news.
`uucp`	Messages generated by UUCP.
`covue`	Messages generated by COVUE.
`tape`	Messages generated by the tape system.
`batch`	Messages generated by the CXbatch product.
`cron`	Messages generated by the `cron` program.

`local`*n* Reserved for local definition. *n* can be any number between 1 and 7.

If more than one *facility* is selected for a severity level, separate each facility with a comma. An asterisk indicates all facilities.

level Severity level of the message. This can be one of the following:

`emerg` Panic conditions. These are normally broadcast to all users (highest level).

`alert` Urgent conditions that should be corrected immediately, such as a corrupted system database.

`crit` Critical conditions, such as hard device errors.

`err` General errors.

`warning` Warning messages.

`notice` Not an error condition, but one that should be specially handled.

`info` Informational messages.

`debug` Information of use when debugging a program (lowest level).

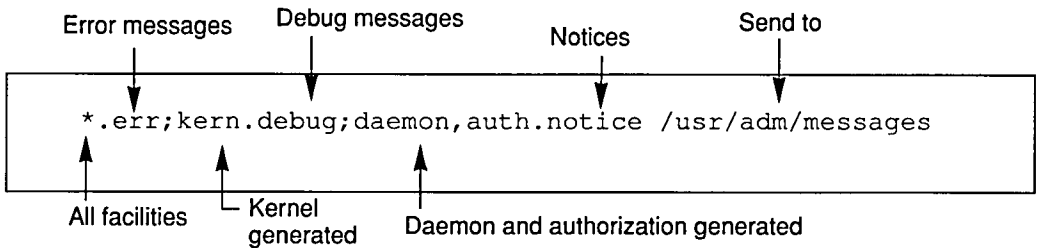
If more than one *facility.level* combination is selected for an entry, separate each combination with a semicolon (;). Selecting a level causes it and all higher levels to be logged.

send_message_here can be one of the following:

- Absolute path name of a file; writes messages to the named file. The file named here must exist before messages can be logged to it. Be sure to complete Step 3.
- Host name preceded with @; forwards messages to the named site.
- A list of users separated by commas. These users receive the messages if they are logged in.
- An asterisk, which sends messages to all users logged on.

For example, the entry shown in Figure 139 indicates that error messages generated from all facilities, debug messages generated by the kernel, and notice messages generated from the authorization system and daemons, are logged to the /usr/adm/messages file.

Figure 139 Example syslog.conf file



See the syslogd(8) man page for more details.

Step 3 If you specified any path names in the *send_message_here* field of the syslog.conf file, create those files using the `touch` command. For example, in Figure 139 messages are sent to the /usr/adm/messages file. Enter:

```
# touch /usr/adm/messages
```

Step 4 Reinitialize the syslog daemon, `syslogd`. Enter:

```
# kill -HUP `cat /etc/syslog.pid`
```

Step 5 Check the /etc/rc.local file to be sure the following lines exist. This line starts the `syslogd` daemon each time the system is booted, as shown in Figure 140.

Figure 140 Example /etc/rc.local file

```
if[ -f /usr/etc/syslogd ]; then
          rm -f /dev/log
          $Ex /usr/etc/syslogd & echo -n ' syslogd'
          if [ -f /usr/bin/X11/execqt -a -f /usr/adm/bin/errlogd ]; then
          $Ex /usr/bin/X11/execqt /usr/adm/bin/errlogd & echo -n '
                    errlogd'
          fi
fi
```

Step 6 If these lines do not exist in the /etc/rc.local file, add them using an editor.

Activating the availability history log file

ConvexOS provides software, called `avail`, that collects system uptime statistics, builds log files with this information, and either electronically mails these files at regular intervals to CONVEX or archives them to tape. Once the `avail` software has been configured, it automatically performs the following tasks:

- Writes a status line to /usr/spool/convex/availlog every 15 minutes. This status line contains:
  - Local time in *ddmmyyhhmmss* format
  - Number of users currently logged in
  - Load average for the last 15 minutes
- Logs information to the /usr/spool/convex/reboot_log file at each multiuser reboot. This data includes:
  - Local time in *ddmmyyhhmmss* format
  - CPU serial number
  - Architecture type of machine
  - Reason for shutdown
- Compares existing SPU files with those from the previous week and records differences every Sunday morning at 00:00:00. CONVEX Computer Corporation uses the output from this comparison to determine what changes were made to the system. The SPU files compared are:
  - /mnt/usr/scn/cop.out
  - /mnt/DIAG_DB_REV
  - /mnt/DIAG_REV
  - /ioconfig
  - /mnt/usr/scn/cop.mem
  - /mnt/usr/lib/softlog
  - /UNIX_REV
  - /mnt/errlog
  - /mnt/usr/ucode/UCODE_REV
- Either electronically mails the differences logged, the reboot_log file, and the availlog file to CONVEX, or archives them to tape each week.

The contents of the /usr/spool/convex/avail.conf file are used to generate a menu of shutdown reasons at reboot. If you choose a reason from this menu, it is added to the /usr/spool/convex/reboot_log file. Reboot waits for two minutes for you to select a response before continuing.

As shipped, the `avail` software is not active. Perform the following steps to configure the system to automatically generate information to the various log files and send the information to CONVEX Computer Corporation.

Step 1 Log in as the superuser.

Step 2 Edit the /usr/lib/.crontab file to add an entry that activates the `avail` utility. `cron` executes commands at specified dates and times according to the instructions found in the /usr/lib/.crontab file.

Figure 141 illustrates a sample /usr/lib/.crontab file that activates the `avail` software. Add the `-t` option to the `avail` command to archive availability data to tape using the `tar` format instead of mailing the data to CONVEX Computer Corporation. A file named /usr/spool/convex/avail.*m.d.y* is produced, where *m* is the current month, *d* is the current day, and *y* is the current year.

Figure 141 Sample /usr/lib/.crontab file

```
0,15,30,45 * * * * /usr/spool/convex/avail
```

Each line of a crontab file represents one activity; each field in this line is 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	can be any number between 0 and 59.
hour	can be any number between 0 and 23.
date	can be any number between 1 and 31.
month	can be any number between 1 and 12.
day	can be any number between 1 and 7, where 1 equals Monday, 2 equals Tuesday, 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 one 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.

See the crontab(5) man page for more details on specifying commands.

Step 3 Edit the /etc/rc.local file so logging is automatically activated each time the system is booted. The /etc/rc.local file is shipped with the following line commented out with the pound sign (#)

```
#/usr/spool/convex/reboot_script < /dev/console > /dev/console
```

Remove the pound sign (#) so logging is automatically started on system boot.

Step 4 Decide whether or not to modify the avail.conf file in the /usr/spool/convex directory. This file is used to generate a menu of shutdown reasons at reboot. Figure 142 illustrates the default avail.conf file.

Figure 142 Default avail.conf file

```
Scheduled maintenance
Power outage
Hardware upgrade
Software upgrade
Crash/Hang
```

If these entries do not satisfy all your needs, you may want to add others to this file using any editor.

Step 5 If you have selected to archive this data rather than mail it to CONVEX Computer Corporation, add the output file to your list for weekly archiving.

Setting up online man pages

13

This chapter describes how to set up and maintain a set of online man pages and associated indexes.

The manual pages, or man pages, are an online repository for information about ConvexOS. They include information ranging from user commands and application programs to data structures and special I/O device files. While many other documents are provided with the ConvexOS system, man pages remain the central piece of documentation.

Man pages are published in hardcopy form as the *ConvexOS Man Pages*. They are also available online through the `man` command. The `man` command displays the contents of man pages, providing quick access to information contained in the *ConvexOS Man Pages*.

For example, to display information on the `chmod` command, enter:

```
# man chmod
```

The `man` command is customizable and can be tailored to your specific site. For complete information about using the `man` command, refer to the man(1) man page and the *ConvexOS Primer*.

Organization of online man pages

Man pages are provided as `nroff` source files and are located in the /usr/man directory. They are arranged in subdirectories (man1 through man8) that correspond to the section of the *ConvexOS Man Pages* in which the hardcopy man pages are found. For example, the /usr/man/man1 directory holds pages for section 1 (commands and applications), where commands such as `more`, `ls`, and `chmod` are found.

A summary of the /usr/man directory is shown below:

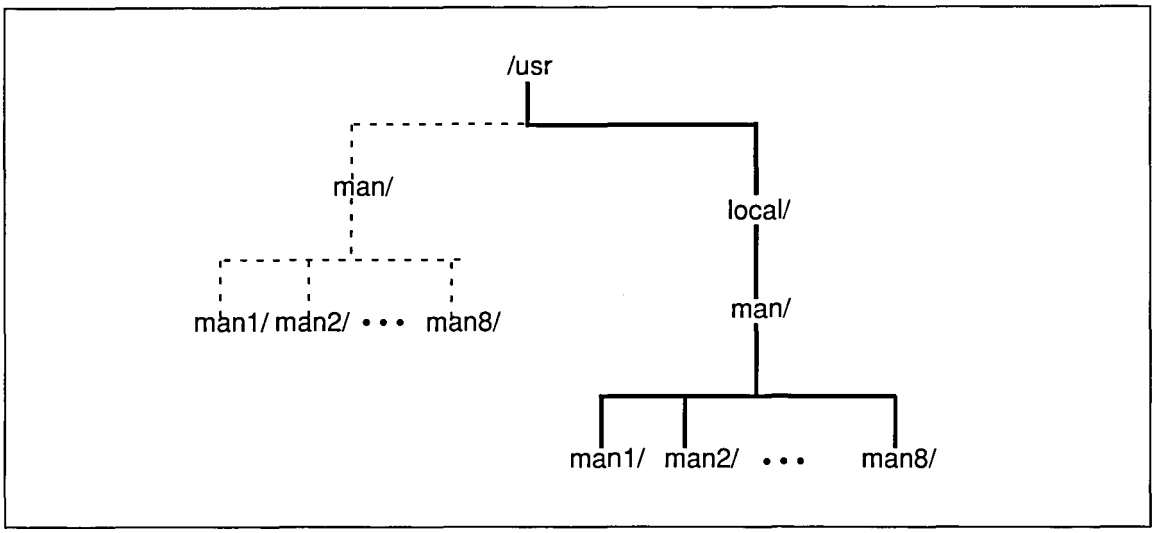
man1	Commands and application programs
man2	System calls
man3	Subroutine libraries
man4	Descriptions of special I/O device files
man5	Structure of system files
man6	Games (not supplied by CONVEX)
man7	Commands for document formatting and code generation macros
man8	System management and maintenance commands
manl	Local man pages
man.template	Template file for creating new man pages.

Typically, each site establishes a set of site-specific utilities and man pages for those utilities. When a new software release is installed on your system, many standard files and directories (for example, /bin and the man1 through man8 directories that store ConvexOS man pages) are overwritten. To make sure that local utilities are not overwritten, place local utilities in the /usr/local/bin directory, and place local libraries in the /usr/local/lib directory. Place locally-generated man pages in the /usr/man/manl directory. In addition to l (local), other subdirectories that have been traditionally used in /usr/man include n (for new man pages), o (for old), and p (for public). While these directories are not provided initially, ConvexOS does support their use through the `man` command. Refer to man(1) for details.

The manl directory is not part of the operating system release and will remain intact on your system.

As an alternative to the manl directory, local man pages can be installed in a separate man tree, /usr/local/man. This is useful if you have a substantial number of local man pages that require greater organization than the /usr/man/manl directory provides. The /usr/local/man directory should be organized similar to /usr/man, where man[1-8] subdirectories provide the same type of organization for local man pages that exist for ConvexOS man pages. This setup is fully supported by the man command and is recommended if you have a large number of local man pages. This will produce a directory tree similar to that shown in Figure 143.

Figure 143 Recommended organization of local man pages



It is also possible to install several complete sets of man pages on the same system using the MANALT environment variable. MANALT/*subdir* specifies the location of alternate sets of man pages, where MANALT is a directory (default is /usr/local/man) and *subdir* is a complete man tree. If you supply two (nonswitch) arguments to the man command, the MANALT directory is consulted to see if there is a subdirectory whose name is the first argument. This makes it easy to request a man page from a specific set of installed man pages. For example

```
% man sun csh
```

would display the csh man page from an installed set of sun man pages.

Formatting online man pages

As stated in the previous section, man pages exist as nroff source files in the /usr/man directory. They are arranged in subdirectories (man1 through man8) that correspond to the sections in the *ConvexOS Man Pages*.

Before the contents of these `nroff` files can be displayed, they must be formatted into readable form. This can be done either individually when the man page is requested, or by preformatting all the man pages using the `catman` utility before they are requested.

With both methods, when the man page is formatted it is saved as a formatted file and stored in a subdirectory (cat1 through cat8) in the /usr/man directory. Each source subdirectory (man1 through man8) has a corresponding format directory (cat1 through cat8).

The cat directories are shipped empty with ConvexOS. Figure 144 lists the cat and man subdirectories shipped with ConvexOS.

Figure 144 Contents of /usr/man directory after executing `catman`

```
# ls -F /usr/man
cat1/  cat4/  cat7/  man2/  man5/ man8/
cat2/  cat5/  cat8/  man3/  man6/ man1/
```

Individually formatting man pages

Each time a user accesses a man page using `man`, the man utility determines whether or not a formatted copy exists in a cat* directory and whether or not it is current. If the formatted version is missing, or if the source file has been modified since the formatted occurred, `man` creates a new formatted page, displays it on the terminal screen, and saves it to the appropriate cat directory. For this to happen, cat* directories must exist and users must have write permission to these directories.

This method saves disk space because you only store formatted copies of man pages that are actually requested by your users.

Preformatting man pages

The catman utility creates preformatted versions of all the man pages that exist in the /usr/man directory. The `catman` utility provides a faster method of displaying information by making formatted man pages available to the user. However, this method takes up a lot of disk space because it stores formatted copies of all the man pages, whether or not they are requested by your users.

To use `catman`, enter:

```
# /etc/catman
```

Note the following about `catman`:

- In addition to formatting man pages, `catman` also creates the /usr/lib/whatis database (unless specifically told not to). Refer to the "Creating a search database" section for more information.
- The `catman` utility should be executed whenever optional products are installed on your system. Installation scripts for CONVEX products automatically place man pages for these products in the /usr/man directory.
- In addition to formatting man pages, `catman` creates the /usr/lib/whatis database (unless specifically told not to.) Refer to the "Creating a search database" section of this chapter for more information.

Creating a search database

The whatis database is a set of files that contain table of contents entries from the NAME section of each man page in the man tree. The files are used by the `man`, `apropos`, and `whatis` commands to view summary information about a given topic. For example, the command

```
# /usr/convex/whatis cat
```

displays the following information on the screen:

```
cat (1) - catenate and print
```

The `makewhatis` utility builds the text and `dbm`[1] forms of the whatis database from online man pages and should be run whenever the NAME section of a man page source file is modified, or when new man pages are installed.

The /usr/lib/whatis database can be created in one of two ways:

- Run the `catman` utility. With no options specified, `catman` formats man pages (as discussed in the previous section) and creates the whatis database by calling the `makewhatis` utility. If you use the `-w` option, only the whatis database is created.
- Run the `makewhatis` utility directly. To use `makewhatis`, enter

  ```
  # /usr/lib/makewhatis
  ```

By default, `makewhatis` builds the database from the /usr/man directory. If you have installed local man pages in /usr/local/man, you must specify the man path with the `-M` option. An example specifying the `-M` option is shown below:

```
# /usr/lib/makewhatis -M /usr/man:\
/usr/local/man
```

The following files are created by the `makewhatis` utility:

/usr/man/whatis	Default whatis database, text version
/usr/man/whatis.pag	`dbm` data file for default whatis database
/usr/man/whatis.dir	`dbm` index file for default whatis database

[1] `dbm` refers to the dbm(3X) database subroutines that are used in creating the whatis database. Refer to the dbm(3X) man page for more information.

Multiple entries in the NAME section or man pages that contain links (hard, soft, or by .so inclusion) are stored under the same man page name. For example, `more` and `page` perform similar functions and are both described on the more(1) man page. The `makewhatis` utility indexes these entries so that they are available online by calling either `more` or `page`. If you enter **`man more`** or **`man page`**, the more(1) man page (which is also the page(1) man page) is displayed.

This method can save a significant amount of disk space because it guarantees that only one page is generated, regardless of how many ways you can access the corresponding man page.

Creating indexes

ConvexOS contains several lengthy man pages, making it difficult to locate specific information quickly. To alleviate this, very long man pages are broken up into several major subsections, creating an index of topics. These topics can be specified with the `man` command, allowing you to go directly to the desired information in that man page. A complete list of available topics can be displayed with the `man -i` command. Refer to the man(1) man page for more specific information on using indexes.

Man page indexes are generated automatically when needed. If an idx* subdirectory exists in the root of the man tree, the index is left there under the same name as its man page for quicker access in the future. The idx* directories must have write permission for all users. Indexes older than their parent man page are rebuilt on demand.

The idx* directories are shipped empty with ConvexOS. Each source subdirectory (man1 through man8) has a corresponding index directory (idx1 through idx8). A listing of the /usr/man directory with idx* subdirectories is shown in Figure 145.

Figure 145 Contents of /usr/man directory after creating index subdirectories

```
# ls -F /usr/man
cat1/ cat6/ idx3/ idx8/ man5/ whatis
cat2/ cat7/ idx4/ man1/ man6/ whatis.dir
cat3/ cat8/ idx5/ man2/ man7/ whatis.pag
cat4/ idx1/ idx6/ man3/ man8/
cat5/ idx2/ idx7/man4/ manl/
```

Granting operator-class privileges

14

At many sites, operators (rather than the system manager) perform routine maintenance tasks such as dumps, restores, tape handling, printer control, and system shutdowns. With ConvexOS, the commands and utilities used to perform these tasks require superuser privileges.

Because superuser privileges provide access to every command and file in the system, you may not want to grant superuser privileges to everyone who may need to perform a maintenance task. The operator interface system, more 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. As shipped, the operator interface is not active. This chapter describes how the operator interface works and how to set it up.

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 to their files only.

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 without receiving full superuser privileges. As system manager, you can restrict:

- **Who may be an operator**—You can restrict an operator class to an individual user, 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 that may be executed by users in an operator class. These commands can include custom scripts or programs that are designed for your site.
- **What arguments the operator may pass to the command**—You can restrict the environment inherited by the commands.

The operator interface system is formed by the op.access file and the `op` utility. The op.access file is located in the /etc directory and contains the rule list for operators and the commands to which they have access. Each line in this file represents one task. The format for the op.access file is

task_name command; operator

where

task_name is the name assigned to the task. This name is used to invoke the task using the `op` facility.

command is the command executed when *task_name* is invoked.

operator is a list of users, groups, or both that can perform the *task_name* using the `op` facility.

For example, the following line in the /etc/op.access file

```
weekly /etc/dump 0Gun /mnt; groups=ops
```

specifies a task called `weekly` that performs a dump of /mnt and designates that members of the group ops can perform the task. The operator performs the task using the `op` utility. To perform the task called `weekly`, enter:

```
# op weekly
```

This is equivalent to entering the following command, except that it does not require superuser privileges:

```
# dump 0Gun /mnt
```

Security issues

Because the `op` utility allows limited operator access to tasks requiring superuser privileges, system security can be breached if the tool is not used wisely. To eliminate this possibility, create the /etc/op.access file with the following considerations:

- Make the file owned by root with 0400 access mode. This way, only the superuser can read the file.
- Do not establish an operator task that calls an interactive utility. Interactive utilities prevent limits on arguments and can often invoke an interactive shell that inherits root privileges from `op`.
- Do not include pagers such as `more` or `less`, or anything else that has a shell escape.
- Be careful in your choice of arguments to commands listed in the /etc/op.access file. For example, if you establish the `restore` command without restricting arguments, a careless operator can destroy file systems by omitting crucial arguments.
- Update the list of users and groups defined in the /etc/op.access file whenever one becomes obsolete, for example, if an operator leaves the company.
- Use syslog to log all attempts to execute `op` and establish logging to ensure that all warning messages receive immediate attention.

Planning the op.access file

Before creating the /etc/op.access file, you must plan what types of tasks should appear in this file. To do this, perform the following steps:

Step 1 Make a list of the tasks requiring superuser privileges that you wish to delegate to users who do not have superuser privileges.

Step 2 Decide which users will perform which tasks and record the decisions on the list created in Step 1. (The superuser can execute any task because access permissions are not checked. Therefore, it is not necessary to include the superuser on your list.)

If several users must perform the same set of tasks, you may want to define them as a group or groups. For example, if several users will perform backups and a different group will manage the line printer queues, you can create two operator groups, such as backoper and printoper.

Step 3 Identify and list the commands necessary to accomplish each task created in Step 1. Use the full command path name; for example, /etc/dump, not just dump.

Step 4 Decide whether you want literal or variable arguments for any commands and include them with the command specified in Step 3. Literal arguments define specific command arguments such as `0Gun`, or specific files such as /dev/rmt20. For example, if you want the task named `weekly` to invoke a dump of the /mnt file system with the `0Gun` command arguments, specify the following command:

```
/etc/dump 0Gun /mnt
```

Variable arguments are specified as `$1`, `$2`, `$3`, ... $*n*, where *n* is a positive integer from 1 to 63. `$1` refers to the first argument given with the `op` command, `$2` refers to the second argument, and so on. Multiple arguments must be separated by spaces or tabs. If an entry contains `$*`, you can specify any number of trailing arguments on the `op` command line, or specify no arguments. For example, the following op.access file entry allows an operator to specify two arguments with the `op` command that invokes the task named `weekly`.

```
weekly /etc/dump $1 $2;
```

Step 5 You can restrict which arguments can be entered by the operator for any variable argument. Separate the variables from the command with a semicolon (;).

Decide on valid values for each variable argument specified in Step 4 and add them to your list. This can be any number of literal values or regular expressions, separated by commas, that take the form:

variable=value[,...]

For example, the second variable (`$2`) in the Step 4 example specifies what file systems to back up. If you want the operator to be able to back up only the /mnt, /tmp, and /usr file systems, you must specify:

```
$2=/mnt,/tmp,/usr
```

Note

Because the /etc/op.access file is not parsed by the C shell, rules for regular expressions differ from those on the command line or in shell scripts. Instead, regular expressions used in strings follow the rules used by `ed`. See the ed(1) man page for more details.

If you do not define limitations for an argument, the operator can enter any value. For example, in the entry in the /etc/op.access file

```
/etc/shutdown -r $1 $2; $1=now,+[0-99],[0-23]:[0-59]
```

the variable argument `$2` is not defined. In this example, any value can be passed to the `shutdown` command for the second argument (`$2`), but only the following expressions can be passed for the first argument (`$1`):

- `now`
- `+00` through `+99`
- `00:00` through `23:59`

Unless you establish an argument as optional, the operator must supply a corresponding value for any variable argument when they invoke the task. To designate a variable argument as optional, define double quotes ("") as one of the possible values for the variable. Then, if the operator wishes to omit the argument, they only need to enter double quotes for the argument on the `op` command line. For example, the following command specifies /mnt, /tmp, or no variables as valid values for `$1`:

```
$1=/mnt,"",/tmp
```

Step 6 Some tasks require specific information to execute properly, such as the root directory path name, the current working directory, or the file-creation mask. Review each task listed in Step 1 and determine whether or not it requires a definition different than the default definition for any of the items listed in Table 31. Add this information to your list.

Table 31 Defaults for command options

Keyword	Value
`chroot`	Specifies the root directory path name that precedes any path name encountered during execution of the task, including the command specified in the op.access file entry. See the chroot(2) man page for more details. Default = current root directory
`dir`	Changes the working directory to the path name specified. Default = current working directory
`egid`	Sets the effective GID to the specified value. The value can be any numeric user ID or login name. Default = real GID value
`euid`	Sets the effective UID to the specified value. The value can be any numeric group ID or group name. Default = real UID value
`gid`	Sets the GID to the specified value. The value can be any numeric user ID or login name. Default = root
`uid`	Sets the UID to the specified value. The value can be a numeric user ID or login name. Default = root
`umask`	Sets the file creation umask to the octal value specified. See Chapter 2, "Security considerations" for more details on setting the umask. Default = 022
`$`*var*	Where *var* is the name of an environment variable. Sets the specified environment variable to the specified value before the command is executed. An item is assumed to be an environment variable if it is an alphanumeric string that begins with a dollar sign ($). If empty, the item passes through with the current value.

Table 31 Defaults for command options (continued)

Keyword	Value
`groups`	Specifies the group name or number or list of group names or numbers separated by commas that can perform the task. Members of a named group are defined in the file /etc/group. If no value is specified for groups, the default list of groups is used. If default groups are not assigned, no groups have access to the function.
`users`	Specifies a user name or list of user names separated by commas that can perform the task. If no value is specified for users, the default list of users is used. If there are no default users assigned, no user has access to this function.
`unsetenv` *$var*	Unsets the environment variable specified on the default line. This allows you to set a default environment variable (such as TERM) that is applicable to most commands, and unset the default value for those commands where it is not applicable.

If a particular entry in the /etc/op.access file does not specify values for these options and it is needed by the command invoked by the entry, the system default value is used. You can change the default by specifying a different default for the item on the default line of the op.access file.

Step 7 Decide on default values for any options listed in Table 31.

Some of the options in Table 31 will have a common definition for multiple entries in the /etc/op.access file. In this case, you can define a default definition for those options in the /etc/op.access file. In this way, you do not have to repeat the common definition multiple times. For example, if users smith, jones, and brown have access to the majority of the entries in the /etc/op.access file, specify them as default users for the users option:

```
DEFAULT users=smith,jones,brown
```

If this is done, any entry that does not define users will allow smith, jones, and brown to perform the task; any entry that defines users ignores the default setting and uses the value specified with the entry. That is, if an entry includes a definition for users such as:

```
users=johnson
```

only user johnson can perform the task; smith, jones, and brown cannot. If you want user johnson and the default users smith, jones, and brown to perform the task, you must specify:

```
users=johnson,smith,jones,brown
```

If you want to deny all users access to a task, leave the value for the groups and users option blank. For example:

```
users=
```

Creating the op.access file

Now that you have decided what types of tasks should appear in the op.access file, what commands to invoke for each task, who can perform those tasks, and default values for various options, you can implement these decisions. To do this, perform the following steps:

Step 1 Log in as superuser.

Step 2 If in Step 2 of the planning phase you decided on groups of operators, add these groups to the /etc/group file using an editor. Each line in this file represents one group; fields in this line are separated by colons. The format for an entry in the group file is

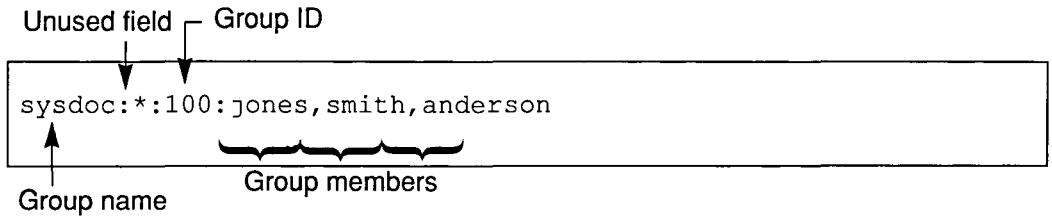
group name:unused field:group ID:group members

where

group name	is the name of the group from 1 to 8 alphanumeric characters long.
unused field	is an unused field and must always contain an asterisk.
group ID	is any unique number between 0 and 32767. When assigning group IDs, assign numbers in sequence. Do not use numbers 0 through 99, as these are reserved for use by CONVEX.
group members	are individual users that are members of the group. Each user name included in the list is separated with a comma. A user can belong to as many as 16 groups.

Figure 146 shows a sample entry in the /etc/group file.

Figure 146 Sample /etc/group entry



Step 3 Open the /etc/op.access file using an editor. If, in Step 7 of the planning phase, on page 280, you decided on default values, define those values on the default line. The default line must be the first noncomment line of the file. The format for the default line is

`DEFAULT` *keyword=value* [,...] [*keyword=value* [,...]]

where *keyword* is the name of the option and *value* can be a single value or a list of values separated by commas. You can specify multiple options on this line, separating each option with a blank space. For example, the following default line specifies group opers as the default group, and /tmp as the current working directory:

```
DEFAULT groups=opers dir=/tmp
```

Step 4 Create an entry in the /etc/op.access file for each task you listed in Step 1 of the planning phase. Each entry in the /etc/op.access file describes the task, the commands used to perform it, and who can perform it. Use the following format

task_name command [*arg* [...]]; [*option* [,...]]

where

task_name is a unique name for an operator task from 1 to 100 alphanumeric characters long. This required field cannot contain spaces or tabs and must begin in column 1.

command is the full path name of the executable command, utility, or script run by `op` when the *task_name* is entered with the `op` command.

arg is any number of literal or variable arguments passed to the command and separated by whitespace. The last argument must be followed by a semi-colon (;).

option is an optional field used to restrict who can perform the task and how it can be invoked. The format for designating an option is shown below:

keyword=value [,...]

where *keyword* is the name of the option and *value* can be a single value or a list of values separated by commas. The possible keywords are listed in Table 31.

Use the following rules when adding entries to the /etc/op.access file:

- An entry can span several lines.
- Separate entries by a blank line for legibility.
- Lines beginning with a pound sign (#) are comments. Everything from the pound sign to the next new line is ignored by `op`.

- If a line begins with white space, it is considered part of the previous line.
- The following characters carry special meaning as field separators or clarifiers; these cannot be used in a string unless they are enclosed in double quotation marks: comma (,), semicolon (;), equal sign (=), dollar sign ($), pound sign (#), or characters in regular expressions (.,[,*).

See Figure 147 for an example of the /etc/op.access file.

Figure 147 Sample /etc/op.access file

```
# the first non-comment line should be the default line; the default
# line specifies the site defaults
#
DEFAULT groups=opers
#
# filesystem backups
weekly /etc/dump 0Gun $1; users=smith,jones,brown $1=/,/usr,/mnt
daily /etc/dump 5Gun $1; users=smith $1=/,/usr,/mnt
#
# take the system down
# $1 shows a good use of regular expressions; $2 can be anything but is
# required
shutdown /etc/shutdown -h $1 $2; $1=now,+[0-9]*,[0-9]:[0-9]*
reboot /etc/shutdown -r $1 $2; $1=now,+[0-9]*,[0-9]:[0-9]*
#
# kill all batch processes so system can be restarted
# (overrides the default group setting, allowing no groups to perform
# task)
killbatch /etc/opbin/kill_batch_procs; groups= users=brown
startbatch /etc/opbin/start_batch;
#
#start up disco daemon
disco /etc/opbin/start_disco; uid=disco gid=proj dir=/scratch
     umask=027 groups=opers,disco users=jones $USER=disco
     $SHELL=/bin/shell
#
# mount and unmount removable drive
rdsmount /etc/mount $1 $2; groups=disco,opers dir=/ users=smith,jones
     $1=/dev/dd0 $2=/.*
```

Step 5 Save and close the /etc/op.access file.

Step 6 Check the syntax for the entries in the /etc/op.access file using the -h argument of the op command:

```
# op -h
```

If the superuser uses the -h help option, op reports all functions in the file, parsing each function and checking it for correct syntax. If the syntax of the /etc/op.access file is correct, op lists each task in the file. If the syntax is not correct, an error message describing the problem is displayed.

If you receive an error, correct the syntax and repeat this step.

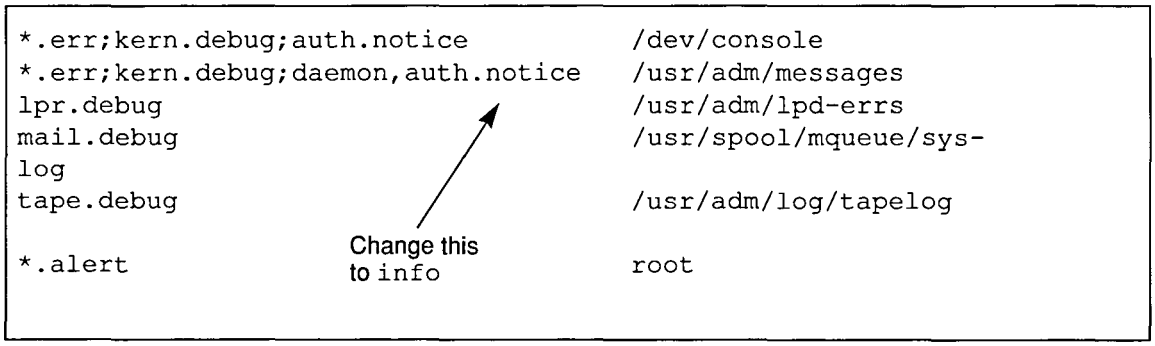
Step 7 Change the access mode of this file to 0400. All users except the superuser should be prevented from reading and writing this file. Enter:

```
# chmod 0400 /etc/op.access
```

Step 8 Modify the /etc/syslog.conf file to log standard usage messages to a message file. Entries in the /etc/syslog.conf file control the type of messages to log and where to log them.

op logs INFO, NOTICE, WARNING, and ERR messages. However, with the /etc/syslog.conf file delivered with ConvexOS, op only logs NOTICE, WARNING, and ERR messages to the console and the /usr/adm/messages file, and discards standard usage messages. Figure 148 illustrates the /etc/syslog.conf file.

Figure 148 Sample /etc/syslog.conf file



Standard usage is logged by op at the INFO level. To get those messages logged to the /usr/adm/messages file, change auth.notice to auth.info on the second line of the syslog.conf file. This logs INFO and all higher level messages to /usr/adm/messages.

Note

op does not log problems with the op.access file; these problems are reported to stderr output.

Step 9 Reinitialize the syslog daemon, syslogd. Enter:

```
# kill -HUP `cat /etc/syslog.pid`
```

The PID associated with `syslogd` can be found in the /usr/etc/syslogd.pid file.

Customizing kernel boot-time parameters

15

When you boot your system, the `boot` command reads a file containing parameters that control the way ConvexOS handles CPUs and CCUs at your site. You can change these parameters to optimize performance and behavior of your system without recompiling the system image. This chapter discusses how to change these parameters, and describes each parameter and its possible values.

Where boot-time parameters are located

There are two files that contain boot-time parameters: the /mnt/os/bootcmd file and the /mnt/os/bootcmd.local file. Both files are located on the SPU disk.

The /mnt/os/bootcmd file is provided with your ConvexOS release tape and contains information about how to boot your system, where the root partition resides, and commands that specify certain system parameters. You should not alter this file, as it is subject to change with each release of ConvexOS. Instead, you should use the /mnt/os/bootcmd.local file to set boot-time parameters specific to your site.

The /mnt/os/bootcmd.local file is not part of the ConvexOS release tape. You must create it to change the default values for kernel boot-time parameters. Commands in the /mnt/os/bootcmd.local file take precedence over those in /mnt/os/bootcmd, allowing you to customize the way your system boots.

Changing parameters

Perform the following steps to create a bootcmd.local file and set boot-time parameters specific to your site:

Step 1 Log in as the superuser.

Step 2 If the bootcmd.local file already exists in the /mnt/os directory on the SPU, copy this file from the SPU to the /tmp directory. Enter:

```
# spu -r /mnt/os/bootcmd.local > /tmp/bootcmd.local
```

Step 3 Using an editor, change existing parameters or add new parameters to this file you just copied to the /tmp directory.

Kernel boot-time parameters are set using the `tune` command. You can specify one parameter for each `tune` command, with each specification on a separate line in the file. Use the format

`tune` *processor parameter=value*

which has the following components:

processor — The processor for which the parameter is specified.

Each kernel boot-time parameter is specific to a system processor and affects only that processor. This could be cpu, hsp, iop, viop.

parameter — The name of the parameter you are changing.

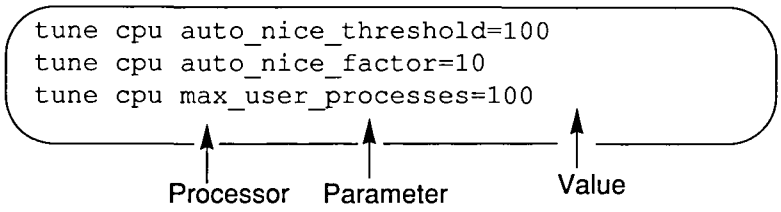
If you are changing a parameter for a CPU, this can be any of the parameters shown in Table 32.

If you are changing a parameter for an IOP, this can be any of the parameters shown in Table 33. Table 33 and Table 34 list VIOP parameters and STREAMS tunables, respectively.

value — The parameter value. This value must be within minimum and maximum values specified for the parameter. The default value, as well as the minimum and maximum values for each parameter are listed in Table 32 and Table 33.

An example of the contents of the bootcmd.local file is shown in Figure 149.

Figure 149 Example bootcmd.local file



Step 4 Copy the modified bootcmd.local file to the SPU. Enter:

```
# spu -w /mnt/os/bootcmd.local < /tmp/bootcmd.local
```

Step 5 Reboot the system. Enter:

```
# shutdown -r +3 "to reboot"
```

Table 32 CPU boot-time parameters

Parameter	Definition
`abspathlen`	Defines the longest path that can exist in a system. The value is not enforced; however, processes that return a pathname return a truncated value for paths that exceed `abspathlen`.
`auto_nice_factor`	Specifies the degree by which the scheduling priority of a process is reduced. This is called renicing. Setting the value to 0 disables renicing. This parameter only affects processes that are not running with the user ID for superuser and that are running at the default nice value (0). See also `auto_nice_threshold` parameter. Default = 4, min = 0, max=64
`auto_nice_threshold`	Specifies the amount of time (seconds of CPU time) before a process is automatically given a reduced scheduling priority (niced). This parameter only affects processes that are running with a nonzero UID and that are running at the default nice value (0). See also `auto_nice_factor` parameter and the getpriority(2) man page. Default = 600, min = 1, max = 9999999

Table 32 CPU boot-time parameters (continued)

Parameter	Definition
`clean_direntry`	Specifies whether or not the name of a file is cleared from the directory entry structure (`struct dirent`) in the kernel upon removal of the file. When a file is removed from a directory, the inode entry is marked as unused in the directory entry stucture, but the file name stored there is not cleared.This is the default setting for `clean_direntry`. When `clean_direntry` is set to 1,the inode entry is marked as unused *and* the file name is cleared upon removal of the file. Default = 0, off = 0, on = 1
`dmon_enable`	Enables kernel event daemons. Set this variable to 1 to enable the daemons required by the CONVEX Storage Manager (CSM). Default = 0, min = 0, max = 1
`dst_algorithm`	Specifies the daylight savings rule used for keeping time on your system. See also `time_zone` parameter and the date(1) man page. Valid values are: 0 = no daylight savings rule 1 = the United States 2 = Australia-style daylight savings 3 = Western Europe 4 = Middle Europe 5 = Eastern Europe 6 = Canada Default = 1, min = 0, max = 6
`enable_unique_core`	Specifies whether each core dump file has a unique name (set to 1) or all core dump files are named core. Setting this parameter prevents one core file from being overwritten by another program that dumps core. Each core file will have a unique name in the form of core.*progname.12345*, where *progname* is the name of the program that dumped core, and *12345* is the PID number of the program when it failed. Default = 0, off = 0, on = 1
`erase_pattern`	Specifies the pattern written over deleted files when the `erase_unlink` parameter is enabled. The default writes binary "1010..." over the deleted files. See also the `erase_unlink` parameter. Default = 0xAAAAAAAA, min=0x80000000, max=x7FFFFFFF

Table 32 CPU boot-time parameters (continued)

Parameter	Definition
`erase_unlink`	Specifies whether deleted files are erased on the disk. When enabled (set to 1), freed disk blocks are overwritten with the pattern specified by the `erase_pattern` parameter. When disabled, erasing does not occur. See also the `erase_pattern` parameter and the "Security considerations" chapter of this book. Default = 0, off = 0, on = 1

`fp_default_mode_issue`	Specifies the system-wide default floating-point mode for programs when no compiler option for floating-point mode is specified. This parameter has no effect on floating-point operations within the kernel and can be overridden by using the compiler command-line option `-f` (format). The value set with this parameter is conveyed to system utilities and user programs through the SI_IEEE_DEFAULT bit of the options word returned by `getsysinfo`.
	If set to 0, programs use native floating-point mode. You can override this specification with the `-fi` option at compile time.
	If set to 1, programs use IEEE floating-point mode if the underlying hardware and software is capable of using it. If the hardware cannot support IEEE floating-point mode, the system prints the following error message at boottime:
	`IEEE floating-point is not available on this system`
	`Default floating-point mode will be NATIVE.`
	You can override this specification with the `-fn` option at compile time.
	Default = 0, off = 0, on = 1

`gateway`	Enables sending Internet Control Message Protocol (ICMP) errors if both of the following conditions are true: • The system has a single network interface, or IP forwarding is disabled (see the `ipforwarding` parameter). • The received IP packet is not for the system that has gateway enabled. If `gateway` is disabled (set to 0) under these circumstances, errors are not sent to the source machine, and the packet is dropped. See also the `ipforwarding`, `ipsendredirects`, and `subnetsarelocal` parameters. Default = 0, off = 0, on = 1

Table 32 CPU boot-time parameters (continued)

Parameter	Definition
`getnewbuf_goal`	Defines the number of buffer headers that are freed at one time. On large memory systems (> 512 Mbytes), the value of 32 does not recycle buffer headers fast enough. Default = 32, min = 32, max = 4096
`ipforwarding`	Enables Internet Protocol (IP) packet forwarding. Packets are forwarded when the IP address does not correspond to any of the Internet addresses for the machine's network interfaces. If this parameter is disabled (set to 0), packets can be dropped. See also the `ipsendredirects`, `gateway`, and `subnetsarelocal` parameters. Default = 1, off = 0, on = 1
`ipsendredirects`	If logging of unsuccessful file access is enabled, this parameter, along with the `logsuspend` parameter, controls logging of unsuccessful file access. Without these parameters, logging continues until there is no more room on the disk. With these parameters, logging is automatically stopped and resumed according to the percentage of disk space available. This parameter specifies the percentage of disk space that must be available for logging to resume after logging is suspended. When there is sufficient free space to allow logging to resume, the following message prints on the console: `File access logging resumed` See also the `log_suspend` parameter. Default = 4, min = 0, max = 100
`logresume`	If logging of unsuccessful file access is enabled, this parameter, along with the `logsuspend` parameter, controls logging of unsuccessful file access. Without these parameters, logging continues until there is no more room on the disk. With these parameters, logging is automatically stopped and resumed according to the percentage of disk space available. This parameter specifies the percentage of disk space that must be available for logging to resume after logging is suspended. When there is sufficient free space to allow logging to resume, the following message prints on the console: `File access logging resumed` See also the `log_suspend` parameter. Default = 4, min = 0, max = 100

Table 32 CPU boot-time parameters (continued)

Parameter	Definition
`logsuspend`	Specifies when to suspend logging unsuccessful file access. When the percentage of free space on the failure log file system drops below the specified percentage, logging is suspended, and the following message prints on the console: `File access logging suspended` You do not enable file logging with this parameter; you enable it with the `faillogon` command. See also the `logresume` parameter. Default = 2, min = 0, max = 100
`max_swapout`	See swap parameters at the end of this table.
`maxregions`	The number of regions a process is allowed. Default = 1024, min = 8, max = 1048576
`maxusers`	This parameter does not directly restrict the number of users on the system. It is used at boot time to size system tables (for example; process, inode, or open file table). If the value is too low, not enough users can be on the system at a time. If the value is too high, memory is wasted. A safe value is the number of tty lines used in your system. If you have a lot of pseudoteletypes, increase the number. If you are regularly running out of processes, increase the system total. See also the `number_ptys` and `number_ttys` parameters. Default = 32, min = 8, max = 256
`max_user_processes`	Specifies the maximum number of processes allowed for each user, including batch jobs. Users running as superuser are not included. Tuning this parameter can help control the load average of the system. A user who tries to start a process beyond the maximum allowed receives the warning message: `No more processes.` Default = 40, min = 4, max = 150
`min_swapout`	See swap parameters at the end of this table.
`miniroot on`	Specifies the partition from which to boot the miniroot. (Load the miniroot to this partition if neccesary.) Default=the b partition of your root disk

Table 32 CPU boot-time parameters (continued)

Parameter	Definition
`nfs_portmon`	Enables Network File System (NFS) server-port checking. The default (0) does not require NFS to check the Internet domain source port of the client machine to see if it is a privileged port. To fully enable NFS server-port checking, you must add the `-p` option for that variable in rpc.mountd file. See the *CONVEX NFS System Manager's Guide* for details. Default = 0, off = 0, on = 1
`nstbuf`	Specifies the number of stripe buffers used for stripe devices. Default = 512, min = 128, max = 8192
`number_ptys`	Specifies the maximum number of pseudoteletype devices. Pseudoteletypes are used by `emacs`, `script`, and some networking programs. Default = 64, min = 0, max = 256
`number_ta_iop_wndw`	Specifies the number of Multibus "windows" the Multibus tape driver will allocate. IOPs contain 256 windows, and the tape driver allocates its windows for direct memory transfers to and from main memory. Increasing this parameter reduces the frequency of tape overruns. Default = 22, min = 8, max = 34
`number_tty_controllers`	Specifies the maximum number of tty controllers. Each controller multiplexes 16 tty lines, so a system with 2 controllers can have up to 32 tty lines connected to the system. This parameter must not exceed the number of controllers licensed to you in your ConvexOS licensing agreement. Default = 2, min = 0, max = 16

Table 32 CPU boot-time parameters (continued)

Parameter	Definition	
pgoutgoal_rssdiv	Used to compute a number of additional pages of a process to page out. Default = 100, min = 0, max = 9999999	The pgoutgoal_rssdiv, pgout_maxrss, and pgout_maxscan parameters are used to compute a number of additional pages of a process to page out in addition to the default number of pages paged. The additional number of pages to page out is calculated by dividing the resident set size of a process in excess of the pgout_maxrss parameter by the value specified in the pgoutgoal_rssdiv parameter.
pgout_macrss	Used to compute a number of additional pages of a process to page out. Default = 1280, min = 0, max = 9999999	
pgout_maxscan	Limits the number of pages of a process that are paged out before another process is selected for paging. Higher values cause more of a process to be paged before the pager continues on to the next process. More pages than specified in this parameter may be paged out because more than one page is picked each time a process is selected for paging. Default = 7680, min = 0, max = 9999999	
sendmsg_access_rights	Enables or disables the sendmsg system call to pass access rights. If disabled, attempts to pass access rights through the sendmsg system call fails with error EACCESS. Default=1, enabled=1, disabled=0	
stripe_devices	Specifies the number of supported striped file systems. The default value of 16 is suitable for most sites. The value should not be significantly higher than the number of striped file systems you are using or intend to use in the near future. Setting this parameter to an unnecessarily high value wastes kernel memory. See the st(4) man page. Default = 16, min = 4, max = 256	

Table 32 CPU boot-time parameters (continued)

Parameter	Definition
`subnetsarelocal`	Considers an Internet address local even if it belongs to another subnet. A different network number means the address is remote. This option is used only during determination of the TCP maximum segment size. TCP uses a small maximum segment size (536 bytes) for remote destinations. For local destinations, TCP uses the maximum transmission unit of the outgoing network interface. See the `ipforwarding`, `ipsendredirects`, and `gateway` parameters. Default = 1, off = 0, on = 1
`suid_shell_script`	Determines the success or failure of attempts to run shell scripts which have the setuid or setgid bit set. When set to zero, attempts to execute shell scripts which have the setuid or setgid bit set fail with errno EPERM. When set to 1, attempts to execute such scripts succeed and the indicated UID or GID is set. When set to two, such scripts are executed, but the UID and GID are not changed, that is, they remain that of the caller of the exec. Default = 0, min = 0, max = 2
`sys_mask`	The default boot-time umask. CONVEX recommends that you do not change this value. Default = 0, min = 0, max = 0777
`ta_force_EOF_on_close`	Controls the writing of end-of-file (EOF) marks when a magnetic tape unit is closed. When a tape unit is closed, the tape driver sometimes writes end-of-tape marks on the tape at the current file position. For historical reasons, the decision about whether to write tape marks depends in part on the access type (read/write) options specified when the tape unit is opened. If the unit was opened for read and write access, the default is that tape marks are only written if the last I/O operation to the tape was a write. If the unit is opened for write-only access, by default tape marks are always written when the tape unit is closed. Turning off this parameter tells the tape driver to write an EOF mark at close time only if the user's last tape I/O operation was a write. Default = 1, off = 0, on = 1

Table 32 CPU boot-time parameters (continued)

Parameter	Definition
`tickadj`	Specifies the rate the system's clock is adjusted (faster or slower) when the `adjtime` command executes. A value of 1 equals a 0.01% change in clock speed, so a value of 100 equals a 1% change in clock speed. Default = 5, min=1, max=1000
`time_zone`	Specifies the number of minutes east (negative values) or west (positive values) of Greenwich Mean Time (GMT). See also the `dst_algorithm` parameter and the date(1) man page. Default = 360, min = -720, max = 720
`tr_nrecs`	Specifies the number of records in the system trace buffer. Setting `tr_nrecs` to 0 disables system tracing. Tuning this parameter is useful primarily during system debugging. Default = 0, min = 0, max = 0x100000
`tty_iop_size`	Specifies the size of IOP tty structures in half-page intervals. tty structures are mapped into main memory through Multibus "windows." Each window can map one page of main memory; windows are allocated to map 16 tty structures for each Multibus tty controller. IOPs contain 256 windows. Increasing this number can reduce the frequency of tty overruns. Default = 1, min = 1, max = 12
`tty_pty_size`	Specifies the size of pseudoterminal (pty) tty structures in half-page intervals. tty structures are allocated in main memory. The number of pty tty structures allocated is defined by the `number_ptys` tunable parameter. Increasing this number can increase the performance of ptys. Default = 2, min = 1, max = 12
`tty_viop_size`	Specifies the size of VIOP tty structures in half-page intervals. tty structures are mapped into main memory through VMEbus "windows." Each window can map one page of main memory; windows are allocated to map 16 tty structures for each VMEbus tty controller. VIOPs contain 1024 windows. Increasing this number can reduce the frequency of tty overruns. This number should be tuned to a multiple of two. Default = 2, min = 2, max = 12

Table 32 CPU boot-time parameters (continued)

Parameter	Definition
`updcksum`	Enables check-summing of User Datagram Protocol (UDP) datagrams. UDP check-summing incurs substantial overhead because each transmitted and received UDP datagram is check-summed. Turning the parameter off increases the risk of allowing bad packets farther up in the protocols. Default = 0, off = 0, on = 1
`viop_enet_proc`	Specifies the number of send and receive processes allowed for each VIOP channel of an Ethernet board at any given time. Set this parameter if the system is unable to boot because of lack of memory. The format for this setting is `0x`*nnnn* The last number represents the first VMEbus Ethernet controller specified in the /ioconfig file, the second to the last number represents the second VMEbus Ethernet controller specified in the /ioconfig file, and so on. For example, 0x1124 specifies that the first VMEbus Ethernet controller in the /ioconfig file can have 4 send and receive processes, the second can have 2 processes, the third and fourth can have 1. Default=0x4444, min=0x1111, max=0x4444

Table 32 CPU boot-time parameters (continued)

Parameter	Definition	
`max_swapout`	The maximum time in seconds a process must be swapped out before it is eligible to be swapped back in. Default = 300, min = 0, Max = 9999999	The `max_swapout`, `min_swapout`, and `swap_pagerate` parameters determine when a process is eligible to be swapped in. To determine eligibility, the system divides the number of pages swapped for the process by the value specified for the `swap_pagerate` parameter, and adds this number to the `min_swapout` value. If the time the process has spent swapped out exceeds the result, the process is eligible to be swapped back in.
`min_swapout`	The minimum time in seconds a process can be swapped out before it is eligible to be swapped back in. Default = 0, min = 0, Max = 9999999	
`swap_pagerate`	The number of pages per second to be divided into the number of pages swapped out for the process to determine how long a process will take to swap in. Default = 64, min = 1, Max = 9999999	

Table 32 CPU boot-time parameters (continued)

Parameter	Definition	
`swap_nicehg`	The value specified for this parameter is multiplied by the nice value of the process and the result is added to the priority. (Note that this decreases the priority of processes with negative nice values.) Default = 5, min = 0, max = 9999999	These parameters are used to calculate a priority for swapping processes out. A high numerical priority number increases the chances of getting selected to be swapped out. The priority is completely determined by the `swap_nicechg`, `swap_partswpchg`, `swap_restimechg`, and `swap_rsschg` parameters.
`swap_partswpchg`	The value specified for this parameter is added to the priority for partially swapped processes. This favors swapping more of a partially swapped process. Default = 100, min = 0, max = 9999999	
`swap_restimechg`	The value specified for this parameter is multiplied by the resident time of the process and the result is added to the priority. This is only done for completely loaded processes (that is, ones that are not partially swapped out already). Default = 1, min = 1, max = 9999999	
`swap_rsschg`	The resident set size of the process is divided by the value specified for this parameter, and the result is added to the priority. Default = 16, min = 1, max = 9999999	

Table 32 CPU boot-time parameters (continued)

Parameter	Definition
`Accelerate_enable`	Specifies whether or not to use a hardware enhancement that accelerates data throughput. To use this enhancement, you must have Revision E or later IOP hardware. If you enable this parameter and do not have the required hardware, the request to use accelerate mode is ignored. Default = 1, off = 0, on = 1
`ca_timer_code`	Controls the frequency with which the ACM-001 or ACM-002 terminal controllers interrupt the IOP. The optimal value for this parameter is site-dependent and is affected by • Controller model • Number of active ports • Baud rate(s) • Other peripherals • General system load The default setting is sufficient for most system loads. In rare cases, high-speed tty input can cause data-overrun errors. If this happens, increase the frequency of IOP interrupts to minimize overruns. A bad value for this parameter can severely degrade system performance. Contact the TAC for more information before tuning this parameter. Default = 8, min = 0, max = 15

Table 33 VIOP boot-time parameters

Parameter	Definition
`uv_num_windows`	Specifies the number of windows the UltraNet driver will allocate on the VIOP. Default = 512, min = 128, max = 960

Table 34 STREAMS tunable parameters

Parameter	Definition
str_n_queue	Maximum number of queues; default = 0, min = 0, max = 4096
str_n_stream	Maximum number of Streams; default = 0, min = 0, max = 2048
str_n_event	Maximum number of event cells (for STREAMS bufcalls); default = 0, min = 0, max = 512
str_n_tevent	Maximum number of timeout cells (for STREAMS time-outs); default = 0, min = 0, max =512
str_dblk_4096	Maximum number of 4096-byte datablocks; default = 0, min = 0, max = 128
str_dblk_2048	Maximum number of 2048-byte datablocks; default = 0, min = 0, max = 128
str_dblk_1024	Maximum number of 1024-byte datablocks; default = 0, min = 0, max = 128
str_dblk_512	Maximum number of 512-byte datablocks; default =0, min = 0, max = 128
str_dblk_256	Maximum number of 256-byte datablocks; default = 0, min = 0, max = 128
str_dblk_128	Maximum number of 128-byte datablocks; default = 0, min = 0, max = 128
str_dblk_64	Maximum number of 64-byte datablocks; default = 0, min = 0, max = 128
str_dblk_16	Maximum number of 16-byte datablocks; default = 0, min = 0, max = 128
str_dblk_4	Maximum number of 4-byte datablocks; default = 0, min = 0, max = 128
str_dblk_0	Maximum number of 0-byte datablocks; default = 0, min = 0, max = 128
str_n_mblk	Maximum number of message blocks; default = 0, min = 0, max = 8192
str_lo_pct	Percentage at which a BPRI_LO `allocb` will fail; default = 60, min = 0, max = 100
str_med_pct	Percentage at which a BPRI_MED `allocb` will fail, default = 80, min = 0, max = 100
str_n_muxlink	Maximum number of links (lower multiplexor connections); default = 100, min = 0, max = 512
str_n_push	Maximum number of modules (I_PUSH ioctls); default = 100, min = 0, max = 512
str_msg_sz	Maximum STREAMS data message size (in bytes); default = 4096, min = 0, max = 65536
str_ctl_sz	Maximum STREAMS control message size (in bytes); default = 4096, min = 0, max = 65536

Generating system images 16

System generation is the process of creating new ConvexOS images that run on the CPU and the channel control units (CCUs). As it is shipped on your CONVEX system, ConvexOS is suitable for most system configurations. However, you must generate a new operating system image if you:

- Install user-written device drivers. Refer to the *CONVEX Guide to Writing Device Drivers*.
- Install layered products with special device drivers, such as the CONVEX UltraNet Interface and OSI WAN.
- Install ConvexOS kernel patch code (if the patch is an entire module, not just an `adb` patch.)
- Install a custom version of ConvexOS. You must have a source license to do this.
- Install products from CONVEX Special Systems.

Minor changes to system configurations or system configuration files may have unexpected results. If you decide to change your system configuration, contact the CONVEX Technical Assistance Center (TAC) to discuss proposed changes and the consequences and methods of implementing them.

System generation configuration file

System generation creates a new version of the operating system based on the specifications of your system configuration file. The system configuration file contains two sections that specify:

- System parameters (some of which have user-selectable options)
- Hardware device types

Several sample configuration files are shipped with ConvexOS: REL_C1, REL_C2, and REL_C3. These sample configuration files are configured for all hardware device types that are supported on standard CONVEX systems. Figure 150 illustrates the first section (system parameters) of the example system configuration file REL_C1.

Figure 150 System configuration file: system parameters

```
machine          c1
cpu              "C-1"
ident            rel_c1
maxmemsize       512
spus             c1
options
NFS,NFSCLIENT,SECURE_NFS,TRACE,INET,QUOTA,NOSEMA,UNET,_ACL,AUDIT,Secur
eWare
pseudodevice     nfs 1
pseudodevice     inet 1
pseudodevice     loop 1
pseudodevice     ether 8
pseudodevice     nc 1
pseudodevice     unet 64
source           yes
config           vmunix root on da0 swap on da0 and da1 and da2 and da3
and da4 and da5 and da6 and da7 and da8 and da9 and da10 and da11 and
da12 and da13 and da14 and da15 and dd0 and dd1 and dd2 and dd3 and dd4
and dd5 and dd6 and dd7 and dd8 and dd9 and dd10 and dd11 and dd12 and
```

The system configuration file has several types of parameters:

- Configuration parameters (lines beginning with the keywords `machine`, `cpu`, `ident`, `maxmemsize`, and `source`)

- System options (the line beginning with the keyword `options`)
- Pseudodevices (lines beginning with the keyword `pseudodevice`)
- The config line (beginning with the keyword `config`)

You may include comments on any of the specification lines. The comment must not exceed one line and must be preceded by a # symbol.

From this system configuration file and files in the /sys/sysgen directory, the `sysgen` utility creates the files and directories necessary to build a system.

Note

Throughout this chapter, the name used for the system configuration file is GENERIC.

Generating a system image

Perform the following steps to generate a new system image:

Step 1 Log in as the superuser.

Step 2 Create a system configuration file in the /sys/sysgen directory that accurately specifies global system parameters and hardware specifications for your site. CONVEX ships three example system configuration files that may be used as templates for custom system configuration files:

- /sys/sysgen/REL_C1, used for C1 Series CONVEX systems
- /sys/sysgen/REL_C2, used for C3200 Series CONVEX systems
- /sys/sysgen/REL_C3, used for C3400 and C3800 Series CONVEX systems

Caution

Do not modify the section of the system configuration file that specifies hardware device types unless you are adding user-written device drivers. In this case, see the *CONVEX Guide to Writing Device Drivers* for more information before creating the system configuration file.

Select the sample system configuration file that corresponds to the system you wish to build (C1 Series, C200 Series or C3 Series) and copy that file to a new file name in the same directory. By convention, the file name is uppercase. For example, copy the file /sys/sysgen/REL_C2 to /sys/sysgen/GENERIC with the following command:

```
# cp /sys/sysgen/REL_C2 /sys/sysgen/GENERIC
```

While there are no restrictions on the file name of the system configuration file, it should be meaningful to the system manager (such as the uppercase version of the system host name).

Step 3 Using an editor, set configuration parameters in the configuration file. You must set parameters:

- `ident`
- `source`

You must not change parameters:

- `cpu`
- `machine`

- `maxmemsize` (You may change this parameter only if your site has a source license.)
- `spus`

Each configuration parameter is listed on a separate line and defines a characteristic of the system.

Figure 151 illustrates configuration parameters in the system configuration file.

Figure 151 System configuration file: configuration parameters

```
machine         c1
cpu             "C-1"
ident           rel_c1
maxmemsize      512
source          yes
```

The format for configuration parameters in this file is

parameter value

where

parameter is the keyword naming the configuration parameter. This can be:

`cpu`	Specifies the CPU type. The value must be enclosed in double quotation marks.
`ident`	Identifies the system by the name of the system configuration file. By convention, this file name is uppercase.
`machine`	Specifies the system type. Currently, the only supported values are `c1`, `c2`, `c3` and `convex`.
`maxmemsize`	Specifies the number of megabytes of memory supported by the vmunix image.
`source`	Specifies whether you have a source license for ConvexOS or a binary license. The values are `yes` and `no`.
`spus`	Specifies the type of SPU. May be `c1`, `c2`, `c34`, or `hp332`.

value is the option that you specify for the configuration parameter.

Step 4 Using an editor, set system options in the configuration file. You must have ConvexOS source code for changes to the system options to have any effect. Figure 152 illustrates the `options` parameter in a system configuration file.

Figure 152 System configuration file: system options

```
options  NFS,TRACE,INET,QUOTA,NOSEMA,NFSCLIENT,UNET
```

The format for options in this file is

```
options [value,...]
```

where

`options`	is the keyword for the line specifying system options.
value	is a system option. Several values may be listed on a single line separated by commas, or each value may be specified on a separate options line. You can specify one or more of the following:
`NFS`	Supports Network File System
`TRACE`	Supports kernel trace points
`INET`	Supports Internet communications protocol
`QUOTA`	Supports disk quotas
`NOSEMA`	Omits conditionally compiled kernel semaphoring
`NFSCLIENT`	Adds additional support for NFS
`UNET`	Supports UltraNet

The `sysgen` utility creates a makefile that causes the flag `-Dvalue` to be passed on the `cc` command line for each source file that is compiled. However, compiling the kernel with a particular system option enabled in the system configuration file does not ensure that the option will be enabled. The `TRACE` option requires a patch to the kernel before the option can be used. CONVEX compiles all supported options except `TRACE` into the system images and libraries that are shipped with the standard release.

Step 5 Using an editor, modify pseudodevices in the configuration file. You must have ConvexOS source code for changes to the pseudodevices to have any effect.

Pseudodevices are drivers and software subsystems that are treated like device drivers but do not have any associated hardware. To include pseudodevices in your system, you must specify the name of the device and the number of devices on your system in the system configuration file.

Each pseudodevice must be specified on a separate line. Figure 153 illustrates pseudodevice specifications in a system configuration file.

Figure 153 System configuration file: pseudodevices

```
pseudodevice     nfs 1
pseudodevice     inet 1
pseudodevice     ether 8
```

The format for pseudodevices in this file is

`pseudodevice` *device_name number*

where

`pseudodevice`	is the keyword for the line specifying a pseudodevice.
device_name	is the name of the pseudodevice. This can be one of the following:
`nfs`	Is required to support the Network File System (NFS). See the *CONVEX Network File System System Manager's Guide.*
`loop`	Specifies the software loopback interface.
`inet`	Specifies DARPA Internet protocols.
`ether`	Is used by the Address Resolution Protocol on 10 Mb/s Ethernets. Must be greater than or equal to the number of Ethernet controllers in the system.
`nc`	Specifies COVUEnet.
`unet`	Specifies UltraNet.
number	is the number of pseudodevices on the system.

The /sys/sysgen/pseudo_devices file lists all supported pseudodevices. The `sysgen` utility reads the pseudo_devices file and uses it to validate pseudodevice names in the system configuration file and create header files for the pseudodevices.

Step 6 Using an editor, modify the `config` line in the configuration file.

The system configuration is specified on a single line in the system configuration file beginning with the keyword `config`. Figure 154 illustrates the config line in a system configuration file.

Figure 154 System configuration file: config line

```
config vmunix root on da0 swap on da0 and da1
```

The format for the config line in this file is

`config` *kernelname configuration_clause* `[...]`

where

`config`	is the keyword.
kernelname	is the name of the CPU system image. The default is `vmunix`.
configuration_clause	a clause that specifies a configuration value. This can be one or more of the following separated by spaces:
`root on` *root_device*	where *root_device* specifies the location of the root file system.
`swap on` *swap_device* [`and` *swap_device*]	where *swap_device* specifies the paging and swapping areas.

In the example in Figure 154, the root file system is on partition *a* of da0 (`a` is the default partition for the root file system). Swapping is specified in partitions *b* of da0 and da1 (`b` is the default file partition for swap). Specifying two partitions for swapping means that partitions da0b and da1b are interleaved.

By convention, the *b* partition of a disk is used for swapping. If the system tries to swap on a partition that contains user data, that data is destroyed. If the swap partition /dev/da0b does not exist when you install a new version of ConvexOS, the system will not boot.

Device names may be fully specified (that is, listing device, unit, and partition) or specified only by device and unit number, in which case the `sysgen` utility selects default partitions.

Note

These parameters can also be modified by using tunable parameters. Refer to Chapter 15, "Customizing kernel boot-time parameters, on page 287, for more information.

The instructions in the procedure below assume that you are generating the vmunix system image for the CPU and all images (hsp, iop, and viop) for the CCU processors (HSP, IOP, VIOP). However, you do not need to execute commands for CCU processors that are not installed on your system, and you need only execute commands for the CPU or a CCU processor if you:

- Changed the source code for that processor.
- Specified system configuration parameters for that processor.

If you are not sure whether or not changes you made affect a particular processor, execute the commands for that processor when performing system generation; if you did not make changes, executing these commands simply regenerates the current system image.

Step 7 The following steps assume that you are running the C shell on the system console and logged in as root. If your default shell is the Bourne shell, change to a C shell by entering:

```
# csh
```

Step 8 `sysgen` must be run from the /sys/sysgen directory. Change to the /sys/sysgen directory by entering:

```
# cd /sys/sysgen
```

Step 9 Execute the `sysgen` utility by entering:

```
# ./sysgen GENERIC
```

The `sysgen` utility creates the following directories in the /sys directory to hold object-code files:

- /sys/GENERIC
- /sys/GENERIC_hsp
- /sys/GENERIC_iop
- /sys/GENERIC_viop
- /sys/GENERIC/os
- /sys/GENERIC/sysgen

In each of the first four directories, `sysgen` creates a makefile listing program and file dependencies for either the vmunix system image or a CCU system image.

In /sys/GENERIC/sysgen, `sysgen` creates:

- Header files (with .h suffixes) defining the devices that are compiled into the system

- A header file (with a _conf.h suffix) listing driver entry points

The fifth directory, /sys/GENERIC/os, holds system images generated by the `make` utility (one of the steps in the procedure).

The `sysgen` utility also creates the file /sys/GENERIC/sysgen/swap.h, which describes the location of the root and swap partitions. Information for this file is derived from the configuration file.

Step 10 If `sysgen` finds errors in any file that it uses, it prints an error message. Appendix A, "`sysgen` error messages", lists and explains possible messages. If an error occurs, correct the error and execute `sysgen` again before proceeding.

Step 11 The command:

```
# make depend install >& make.out
```

- Creates a list of dependencies that determine code and data files that must be compiled
- Compiles and links system files
- Installs binary files in the execution directory

Run this command for each system generation directory. You must issue the command in the directory. Enter the following series of commands to generate the desired files in each required directory. (You may skip a directory if you know it was unaffected by changes you made to source code or to system configuration parameters or if you do not have that particular processor on your system.)

```
# cd /sys/GENERIC_hsp
# make depend install >& make.out
# cd /sys/GENERIC_iop
# make depend install >& make.out
# cd /sys/GENERIC_viop
# make depend install >& make.out
# cd /sys/GENERIC
# make depend install >& make.out
```

Step 12 Log in as the superuser on the system console.

Step 13 When the bootable system-image files have been created, they must be moved to the SPU disk, from which the new operating system is booted. The current system image files on the SPU will be overwritten when the new bootable system image files are copied to the SPU. Make backup copies of the existing files before this happens. To do this, shift to SPU OS by pressing CTRL-P.

Step 14 Change to the /mnt/os directory by entering:

```
(spu)> cd /mnt/os
```

Step 15 Ensure that you have at least 3 Mbytes of free disk space in the /mnt/os directory:

```
(spu)> df /mnt
```

Step 16 Make back-up copies of the existing system image files you have created on the SPU by entering the following series of commands that apply. (You may wish to skip files if you know they were unaffected by changes you made to source code or to system configuration parameters or if you do not have that particular processor on your system.)

```
(spu)> mv vmunix vmunix.save
(spu)> mv hsp hsp.save
(spu)> mv iop iop.save
(spu)> mv viop viop.save
(spu)> mv idc idc.save
```

Step 17 Exit the SPU by pressing CTRL-D.

Step 18 Copy each new system-image file to the SPU by entering the following commands or the files you have created. (You may wish to skip files if you know they were unaffected by changes you made to source code or to system configuration parameters or if you do not have that particular processor on your system.)

```
# /usr/convex/spu -w /mnt/os/vmunix < /sys/GENERIC/os/vmunix
# /usr/convex/spu -w /mnt/os/hsp < /sys/GENERIC/os/hsp
# /usr/convex/spu -w /mnt/os/iop < /sys/GENERIC/os/iop
# /usr/convex/spu -w /mnt/os/viop < /sys/GENERIC/os/viop
# /usr/convex/spu -w /mnt/os/idc < /sys/GENERIC/os/idc
```

Step 19 Shut the system down to the SPU by entering:

```
# shutdown -h +5 "rebooting new kernel"
```

Step 20 Boot to single-user mode by entering:

```
(spu)> boot single
```

Messages are printed to the screen. The boot is complete when the system prompt appears.

Step 21 Verify the integrity of the file system by running `preen`

```
# preen
```

Information about the file systems is printed to the screen. `preen` is complete when the system prompt returns.

Step 22 Boot to multiuser mode by pressing CTRL-D.

Configuration file grammar

System generation is complete and the new ConvexOS operating system is installed.The grammar example shown in Figure 155 is a compressed form of the actual `yacc` grammar used by `sysgen` to parse configuration files. Terminal symbols are shown all in uppercase, literals are in **bold type**; optional clauses are enclosed in brackets ([and]); and, zero or more instances are denoted with an asterisk (*).

Figure 155 Compressed example of `sysgen` configuration file grammar

```
Configuration ::=  [ Spec  ;  ]*
Spec ::= Config_spec
      |  hardware  CCU_spec [CCU_spec]*
      |  trace
      | /* lambda */
/* configuration specifications */
Config_spec ::=   machine  ID
      |  cpu  ID
      |  options  Opt_list
      |  ident  ID
      | System_spec
      |  source  yes_no
      |  pseudodevice  ID NUMBER
/* system configuration specifications */
System_spec ::=  config  ID System_parameter [ System_parameter ]*
System_parameter ::=  swap_spec | root_spec
swap_spec ::=   swap  [  on  ] swap_dev [  and  swap_dev ]*
swap_dev ::=  PARTITION_NAME [  size  NUMBER ]
root_spec ::=   root  [  on  ] PARTITION_NAME
yes_no ::=  yes  |  no
/* option specifications */
Opt_list ::=  Option [  ,  Option ]*
Option ::=  ID [  =  Opt_value ]
Opt_value ::=  ID | NUMBER
CCU_spec ::=   ccu  NUMBER  type IOP  Multibus_spec [Multibus_spec]*
      |  ccu  NUMBER  type HSP  Driver_spec [Driver_spec]*
      | ccu  NUMBER  type VIOP Viop_spec [Viop_spec]*
Multibus_spec ::=   multibus  NUMBER Controller_spec [Controller_spec]*
Viop_spec ::= vme NUMBER Controller_spec [Controller_spec]*
Controller_spec ::= controller type ID at csr NUMBER int NUMBER \
Unit_sp[Unit_spec]*
Unit_spec ::=   unit  NUMBER  type  ID |  unit  NUMBER - NUMBER  type  ID
Driver_spec ::=  driver  ID  csr  NUMBER Channel_spec [Channel_spec]*
Channel_spec ::=  channel  NUMBER  type  ID
```

Lexical conventions

Terminal symbols are loosely defined as:

ID	One or more alphabetic characters, either uppercase or lowercase, and underscore (_).
NUMBER	Information about the C language specification for an integer number. That is, a leading "0x" signifies a hexadecimal value, and a leading "0" signifies an octal value; otherwise, the number is interpreted as a decimal value. Hexadecimal numbers may use either uppercase or lowercase alphabetic characters.
PARTITION NAME	The name of a disk partition, such as da0a, or a disk drive, such as da0. When a drive, rather than a partition, is specified, `sysgen` picks a default partition on that drive.

Comments in configuration files begin with a "#" character; the remainder of the line is discarded.

A specification is interpreted as a continuation of the previous line if the first character of the line is a tab.

Configuring the `contact` utility

17

The `contact` utility is an online system for reporting problems to the CONVEX Technical Assistance Center (TAC). This chapter explains how `contact` can be configured to:

- Deliver problem reports via UUCP
- Deliver problem reports via a network
- Deposit problem reports in a single location, so they may be put on a tape and sent to the TAC

For information on using `contact` to report problems, refer to Appendix D, "Reporting problems."

The contactcap file

The system configuration file for `contact` is /usr/lib/contactcap. It contains a series of colon-separated fields that describe:

- Local options
- Delivery

Figure 156 shows the contents of the default /usr/lib/contactcap file.

Figure 156 Default /usr/lib/contactcap file

```
% cat /usr/lib/contactcap
c0|contact|contact site configuration:\
        :ph=(800) 952-0379:ed=/usr/ucb/vi:mb=contact:\
        :cc=%s contact-reports:ta:pa=convex!:ml#10000:ul#50000:
```

Table 35 lists the /usr/lib/contactcap fields.

Table 35 Fields in the /usr/lib/contactcap file

Name	Type	Default	Description
`cc`	string	%s	Carbon copy list; %s expands to the name of the user who submitted the report. Names must be separated by blanks.
`ed`	string	/usr/ucb/vi	Default text editor.
`em`	string	none	Message displayed upon submission of a report.
`mb`	string	contact	Mailbox to send report to.
`ml`	number	10000	Mail file size limit in bytes.
`nh`	string	none	Ethernet host.
`pa`	string	convex!	UUCP path name to CONVEX.
`ph`	string	(800) 952-0379	CONVEX TAC phone number.
`rt`	boolean	false	`contact` may be invoked by superuser.
`ta`	boolean	false	Site has a tape drive.
`tn`	string	none	Technical assistant center name.
`ul`	number	50000	UUCP file size limit.

The default contactcap file shown in Figure 156 is minimal and may not be appropriate for your site. The following sections describe additions and changes you may want to make to /usr/lib/contactcap.

Setting local options

The following /usr/lib/contactcap fields control local options:

`ed` This field contains the path name for the text editor that is automatically be invoked when users choose to edit a `contact` report. By default, it is /usr/ucb/vi unless the EDITOR environment variable is set.

`em` This field contains the message that will be displayed after a user submits a report. By default, there is no message.

`rt` If this field is specified, the superuser may invoke `contact`. It is recommended that you not specify this field; reports sent by the superuser do not have a real username and it will be difficult for the CONVEX Technical Assistance Center to contact the submitting user directly.

`ta` If your site has a tape drive, this field should be specified; if you do not have a tape drive, this field should not be specified.

`cc` This field contains the carbon-copy list for `contact` reports. By default, a carbon-copy of a contact report is sent only to the submitting user (indicated by %s) and to the contact-reports alias. contact-reports is a standard alias in /usr/lib/aliases; by default this alias contains no real users. For more information on aliases, see the chapter "Setting up `sendmail`" or the aliases(5) man page.

If you choose to add additional user names to the `cc` field, they must be separated by blanks.

`tn` This field contains the name of the CONVEX Technical Assistance Center. This information is only displayed when a problem occurs while running `contact`. You may wish to replace this with the name of a local system administrator or user support specialist. If you change this field, you must also change the `ph` field to contain an appropriate phone number.

`ph` This field contains the phone number of the CONVEX Technical Assistance Center, which is displayed whenever the contents of the `tn` field are displayed. If you have changed the `tn` field to include the name of a local system administrator, you should change this field to an appropriate phone number.

Please note that if this field is deleted, it will default to the CONVEX TAC phone number.

Figure 157 contains example local options.

Figure 157 Sample /usr/lib/contactcap local options

```
:ed=/usr/convex/emacs:\
:cc=%s contact-reports pat chris:\
:ta:tn=Pat Smith, System
```

In this example:

- `emacs` is invoked if users choose to edit their `contact` report and do not have a $EDITOR environment variable set.
- Carbon copies of a `contact` report will be sent to the submitting user and to users joe and betty.
- The superuser may not invoke `contact`.
- The site has a tape drive.
- If a problem occurs while running `contact`, users are instructed to contact Pat Smith at extension 4710.

Setting delivery options

The following section describes how to configure `contact` to deliver reports to CONVEX via UUCP. If you are not running UUCP but can deliver mail to CONVEX over a network, please skip to the section "Network delivery." If you cannot send electronic mail at all, but would like to use the `contact` utility to gather problem reports locally, please skip to the section "Local delivery only."

UUCP delivery

The following fields control UUCP delivery:

`uu` This field indicates that you have a UUCP connection. You must include this field as `:uu:`. If this field is missing, users will not be able to run `contact`.

`pa` This field contains the UUCP path name to CONVEX. By default, it contains only "convex!" which indicates a direct UUCP connection. If you do not have a direct connection, you must supply a longer UUCP path name in this field. For additional information on UUCP, refer to Chapter 6, "Setting up a UUCP connection," on page 143.

`mb` This field contains the mailbox to which `contact` reports should be delivered. When the `contact` report is mailed, the contents of this field are inserted after the last "!" in the `pa` field. If you are using UUCP, this field must contain "contact".

`ml` This field contains the maximum size of a mail message, in bytes.

`ul` This field contains the maximum size of a transmitted file, in bytes.

Figure 158 contains a sample /usr/lib/contactcap file configured for UUCP delivery.

Figure 158 Sample /usr/lib/contactcap file for UUCP delivery

```
c0|contact|contact site configuration:\
        :ph=(800) 952-0379:ed=/usr/ucb/vi:\
        :cc=%s contact-reports joe betty:\
        :uu:pa=uunet!convex!:mb=contact:\
        :ml#10000:ul#50000:
```

In this example, `contact` reports will be:

- Delivered via UUCP
- Delivered to uunet!convex!contact

- Limited to 10000 bytes; files included with reports will be limited to 50000 bytes

Network delivery

If your machine is connected via a network to another machine that has a UUCP connection, specify the name of that machine in the nh field, as shown in Figure 159.

Figure 159 Sample /usr/lib/contactcap file for network-to-UUCP delivery

```
c0|contact|contact site configuration:\
        :ph=(800) 952-0379:ed=/usr/ucb/vi:\
        :cc=%s contact-reports joe betty:ta:\
        :uu:pa=uunet!convex!:mb=contact:\
        :ml#10000:ul#50000:nh=convexb:\
```

In this example, a machine named convexb has a UUCP connection. contact reports will be delivered to convexb over a network, and will be delivered to CONVEX from convexb via UUCP.

If your machine has the appropriate network connections to deliver mail directly to CONVEX, you should:

- Delete the uu field.
- Set the mb field to nothing by inserting two double quotes. Do not delete this field.
- Set the pa field to include the entire address needed to deliver mail to the user contact at CONVEX.
- Set the nh field to the name of your machine.

An example of this configuration is shown in Figure 160.

Figure 160 Sample /usr/lib/contactcap for network delivery

```
c0|contact|contact site configuration:\
        :ph=(800) 952-0379:ed=/usr/ucb/vi:\
        :cc=%s contact-reports joe betty:ta:\
        :pa=contact@convex.com:mb="":\
        :ml#10000:ul#50000:nh=convexa:\
```

In this example, contact reports will be addressed to contact@convex.com.

For information on the specific networks to which CONVEX is connected, please contact the Technical Assistance Center.

Local delivery only

If you do not have a network or UUCP connection, you can use `contact` to gather problem reports locally. These reports can be put on a tape and mailed to CONVEX.

To do this, complete the following steps:

Step 1 Create a user named contact. Refer to Chapter 7, "Setting up user accounts," on page 161, for information on creating new users.

Step 2 Edit /usr/lib/contactcap to:

- Include the `uu` field.
- Set the `pa` field to the name of your machine, followed by an exclaimation point (!).
- Set the `mb` field to be the local user contact.

This configuration is shown in Figure 161.

Figure 161 Sample /usr/lib/contactcap for local delivery only

```
c0|contact|contact site configuration:\
        :ph=(800) 952-0379:ed=/usr/ucb/vi:\
        :cc=%s contact-reports joe betty:ta:\
        :uu:pa=convexa!:mb=contact:\
        :ml#10000:ul#50000:nh=convexa:\
```

Step 3 `contact` reports are delivered to the mail file for user contact, /usr/spool/mail/contact. To deliver these reports to CONVEX, use `tar` to put /usr/spool/mail/contact on a magnetic tape and mail the tape to the following address:

Convex Computer Corporation
MS TAC
3000 Waterview Parkway
P.O. Box 833851
Richardson, TX 75083-3851

sysgen error messages

A

This appendix contains a description of error messages and warnings produced by the `sysgen` utility. These messages are listed in alphabetical order. If you encounter error messages that are not described, this may indicate a serious error. In that case, please contact the CONVEX Technical Assistance Center (TAC).

The following conventions are used in describing the messages:

- `%c` expands to a single character
- `%d` expands to a number
- `%s` expands to a character string

`Bad entry in controllers file (%s) - ignored`

An entry (%s) in the /sys/sysgen/controllers file has an invalid format. `sysgen` processes the file, but ignores the invalid line. Refer to the *CONVEX Guide to Writing Device Drivers* for more information on the format of this file.

`Bad entry in units file (%s) - ignored`

An entry (%s) in the /sys/sysgen/units file has an invalid format. `sysgen` processes the file, but ignores the invalid line. Refer to the *CONVEX Guide to Writing Device Drivers* for more information on the format of this file.

`Bad processor type in controllers file (%s) - ignored`

In the /sys/sysgen/controllers file, the processor type field of the line specified by %s is not valid. The only recognized types are D (IDC), I (IOP), H (HSP), P (HIPPi), T (TLI), and V (VIOP). `sysgen` continues to run, but ignores the invalid line.

`Build_controller_table: controller table overflow`

The /sys/sysgen/controllers file specifies too many controller types. This is an internal error and is not something that you can fix. Call the CONVEX TAC.

`Build_unit_table: unit table overflow`

The /sys/sysgen/units file specifies too many unit types. This is an internal error and is not something that you can fix. Call the CONVEX TAC.

`C-1, C-2 and C-3 are the only supported CPU types.`

Something other than C-1, C-2 or C-3 is specified as the CPU type in the system configuration file. C-1 is the correct option for all CONVEX systems running CONVEX operating system V6.2; C-1 or C-2 are available for CONVEX operating system V7.0 or later. C-1, C-2 or C-3 are available for CONVEX operating system V10.0 or later.

`Can't create directory`

The error message printed on the line just before this message specifies the directory and further information about the failure.

`cpu type must be specified`

The line specifying CPU is missing from the system configuration file, for example

```
cpu"C-1"
```

C-1 is the correct option for all CONVEX systems running CONVEX operating system V6.2; C-1 or C-2 are available for CONVEX operating system V7.0 or later. C-1, C-2 or C-3 are available for CONVEX operating system V10.0 or later.

`Defaulting primary swap device to %s`

A swap device is not specified in the config line of the system configuration file. This message does not indicate an error, but informs you which device `sysgen` has selected. (%s expands to a file disk name, such as da0b.) The default swap device is the `b` partition of the disk that contains the root file partition. You can avoid this message by adding the swap specification to the system configuration file.

`Don't forget to run "make depend" in each directory.`

`sysgen` generates this message each time it is run to remind you to perform an important step in system generation. This message does not indicate an error.

`Duplicate "source" keyword; assuming "source no"`

The source line occurs multiple times in the system configuration file. It should appear only once. `sysgen` continues as if you do not have source code.

`Extraneous root device specification`

Extra clause(s) occurs in the system configuration file. Remove the extra clause(s) and rerun `sysgen`.

`Illegal channel type`

An HSP channel type is specified in the system configuration file that did not appear in the file /sys/sysgen/units.

`Illegal controller/driver type %s`

The system configuration file has an entry for a controller (IOP or VIOP) or driver (HSP) type %s that does not exist in the file /sys/sysgen/controllers. This typically indicates that the user is trying to use an incorrect name. Users writing their own device drivers may need to edit the /sys/sysgen/controllers file if they are not using one of the reserved names for user-written device drivers. (Refer to the *CONVEX Guide to Writing Device Drivers*.)

`Illegal processor field %s`

An entry in the /sys/sysgen/controllers file specifies an invalid CCU type. The only types currently supported are D (IDC), I (IOP), H (HSP), P (HIPPi), T (TLI), and V (VIOP).

`Illegal unit type`

A unit is specified in the system configuration file whose name does not appear in the file /sys/sysgen/units. This typically indicates that the user is trying to use an incorrect name. Users writing their own device drivers may need to edit the /sys/sysgen/units file if they are not using one of the reserved names for user-written device drivers. (Refer to the *CONVEX Guide to Writing Device Drivers*.)

`Illogical unit range number specified`

Unit numbers can be specified as a range *n-m* if all units are the same type. This message is printed when *n* > *m* or *n* < zero.

`Illogical unit range specified`

When specifying the list of units attached to a controller, an invalid range of units is specified in the system configuration file.

`Invalid argument to "source" keyword; assuming "no"`

An invalid argument appears on the source line of the system configuration file. The only valid arguments are yes and no. `sysgen` continues as if you do not have source code.

`No root device specified`

A `root on` *device* clause is not specified on the config line of the system configuration file. Add one to the system configuration file and rerun `sysgen`.

`Specify machine type, e.g. "machine convex"`

An invalid machine type is specified on the machine line of the system configuration file. Currently, the supported machine types are `c1`, `c2`, `c3`, and `convex`.

`sysgen: malloc() failed`

An attempt to allocate memory for internal use by `sysgen` failed. This is an internal error. Call the CONVEX TAC.

`Unknown % construct in generic makefile:%s`

`sysgen` combines input from template makefiles and other configuration files to generate makefiles for a system. It uses the skeleton makefile as a template for the contents of the new makefile. The skeleton makefile is copied into the new makefile, except for lines of the form

```
%<string>
```

When `sysgen` sees a line beginning with a percent sign, it looks up <string> in an internal table, and replaces %<string> with some appropriate text. If <string> is not found in the internal tables of `sysgen`, the "unknown % construct" message is printed. This happens only if a user modified the skeleton makefile.

`Unknown option %c`

`sysgen` was invoked with an illegal command line option (%c).

`Usage: sysgen` *sysname*

The *sysname* argument is the name of the system configuration file for which `sysgen` should configure a system.

`Warning: swap defaulted to b partition with root on %s partition`

A nonstandard selection for a root partition without specifying a swap partition caused this warning. By convention, an `a` partition is used for the root file system. You specified some other partition and specified no `swap on device` clause. `sysgen` uses the `b` partition of the root disk specified by %s for the primary swap device. Typically, use the da0a, (or dd0a, du0a) partition for the root partition and the da0b (or dd0b, du0b) partition for the primary swap device.

System files

B

This appendix describes system files requiring periodic maintenance. These files, with the exception of /mnt/errlog on the SPU disk, reside on CPU disks. System files for optional CONVEX products (for example, CONVEX Internet Services and CXBatch) are described in their product documentation.

The following information is provided for each file:

- File format
- File description and maintenance instructions
- Related programs
- Examples
- Caveats and bugs

The termcap(3X) man page contains a detailed explanation of termcap-format files. Refer to this man page as you read the descriptions of /etc/disktab, /etc/gettytab, /etc/printcap, /etc/remote, and /etc/stripecap.

The files included in this appendix are listed in alphabetical order by file name, rather than full path name. For cross-reference, these files are listed on the next page in alphabetical order by full path name.

- /etc/activities
- /etc/actwho
- /etc/disktab
- /etc/fstab
- /etc/gettytab
- /etc/group
- /etc/hosts
- /etc/motd
- /etc/nologin
- /etc/nurc
- /etc/op.access
- /etc/passwd
- /etc/phones
- /etc/printcap
- /etc/pwrestrict
- /etc/rc.local
- /etc/remote
- /etc/stripecap
- /etc/syslog.conf
- /etc/ttys
- /usr/adm/acct
- /usr/adm/batch-acct
- /usr/adm/bill-acct
- /usr/adm/diskuse
- /usr/adm/failure_log
- /usr/adm/log/batchlog
- /usr/adm/lpd-acct
- /usr/adm/stat
- /usr/adm/tp-acct
- /usr/adm/wtmp
- /usr/lib/aliases
- /usr/lib/crontab
- /usr/lib/uucp/L-devices
- /usr/lib/uucp/L-dialcodes
- /usr/lib/uucp/L.cmds
- /usr/lib/uucp/L.sys
- /usr/lib/uucp/USERFILE
- /usr/spool/uucp/ERRLOG
- /usr/spool/uucp/LOGFILE

/usr/spool/uucp/ERRLOG
error log

Format

ASCII single-line entries

Description

The /usr/spool/uucp/ERRLOG file keeps track of problems encountered by the UUCP subsystem. It grows slowly and can be ignored unless a major error causes it to expand significantly.

Example

```
ASSERT ERROR (uux) pid: 3999 (1/26-0:29)
                CAN'T OPEN D.convexBKow2 (0)
ASSERT ERROR (uucico) pid: 274 (7/12-16:27)
PKXSTART ret (-1)
```

This example shows typical errors. Errors usually result from system crashes; files in the process of being sent to or received from remote sites at the time of a system crash are sometimes lost.

The following messsages can appear in the uucp LOGFILE:

`FORGED HOSTNAME (`*name*`)`

`ORIGINATED AT (`*IP_address*`)`

`SHOULD BE (`*name*`)`

Programs

`uucp, uux, uucico, uuxqt`

/usr/lib/uucp/L-devices
UUCP dialout device descriptions

Format

%s %s %s %s %s

Description

This file lists automatic call units (dialout modems) and direct-connect lines for the `uucico` utility. Each line in the file describes one direct-connect line. Some lines describe lines previously described but with differing characteristics (for example, baud rate). The field descriptions are as follows:

1. ACU (automatic call unit) or DIR (direct-connect line)
2. Device in the /dev directory
3. Ignored
4. Baud rate
5. Modem type

Example

```
ACU cua1 cua1 300 vadic
ACU cua1 cua1 1200 vadic
DIR tty2a 0 9600 direct
```

Programs

`uucico`

Caveats

The `cu` and `tip` utilities also use these devices.

/usr/lib/uucp/L-dialcodes
common dialcodes used by UUCP dialers

Format

%s %s

Description

This file maps dialing prefixes in the /usr/lib/uucp/L.sys file into dialable sequences.

Example

```
MCI 6514321,654321
```

In this example of an entry in the L-dialcodes file:

- First seven numbers are the local MCI number
- Comma tells the system to wait for a dial tone
- Last six numbers are the MCI access code

With the system entries set up in this manner, a user may dial an MCI number by specifying "MCI" and the desired number (e.g., MCI2145551212).

Programs

`uucico`

Bugs

Some long distance services (such as Rolm) require trailing digits for accounting; this facility does not solve that problem.

See Also

/usr/lib/uucp/L.sys

/usr/lib/uucp/L.cmds
list of valid remote commands

Format

%s

Description

This file **specifies** the following:

- PATH containing valid remote `uux` commands
- Names of valid commands

Example

```
PATH=/usr/local/bin:/bin:/usr/bin
rmail
nfrcv
nfxmit
ruusend
uncompress
uusend
```

Programs

`uux`

Caveats

Choose the commands you include extremely carefully. Giving remote sites access to commands can create security risks.

/usr/lib/uucp/L.sys
dialing sequences for UUCP for remote hosts

Format

%s %s %s %s %s

Description

This file tells `uucico` how to dial remote hosts. Each line in the file describes one way to access a remote host. Each entry must have at least five fields:

1. Name of remote host; use more lines if there is more than one way to dial the host
2. Times to call the host; for example Any, None, and hybrids (Any1700-2400)
3. ACU (automatic call unit) or DIR (direct connect line)
4. Baud rate
5. Telephone number for ACU lines (modified by L-dialcodes) or device name (for direct connects)
6. Describe (respectively) what the remote system should send to you and what you should send to the remote system. Waiting for a null ("") is the same as waiting for nothing at all. The "waiting for" phrase is a hyphen-separated list of phrases and items to send if the phrase is not received within a time-out period (see example).

Use the following procedure to add a new UUCP site:

1. Replicate a working site's entry.
2. Change the site name.
3. Change the calling times (if necessary).
4. Change the baud rate (if necessary).
5. Change the phone number.

Use this procedure when you are calling another site. If the site is calling you, give the site a unique login name and password; otherwise, there is no audit trail for security.

Example

```
convex1 Any ACU 1200 5551212 "" r
login:-BREAK-login: XYuucp

Password: intruder password: thief
```

The example is on three lines for convenience; in the file, the information is on one line. A remote machine can reply in either uppercase or lowercase (e.g., Login, login).

Programs

`uucico`

Caveats

To debug entries in L.sys, use

```
rm -f /usr/spool/uucp/STST.sitename
/usr/lib/uucp/uucico -f -r1 -ssitename -x4
```

This starts a conversation with site *sitename* and displays the conversation on your terminal as the conversation proceeds.

To use debugging, your UID must be 0 (root) or 1 (daemon).

When a new UUCP site contacts you, put its site name in the L.sys file, whether or not you intend to call it. Each entry in the file must have at least five fields. The following example is an entry for a system you do not plan to call:

```
systemname None DIR 9600 null
```

/usr/spool/uucp/LOGFILE
log of UUCP activity

Format

%s %s (%d/%d-%d:%d-%d) [^(]([^)])

Description

The /usr/spool/uucp/LOGFILE file notes completions of UUCP functions; for example, connects, queues, file copies, and executes. The /usr/spool/uucp/STST.*site* file contains status information; for example, information about recent failed calls and calls in progress. The following is a partial list of LOGFILE entries:

ACCESS (denied)
: In a file-access operation, a file's path prefix might be bad, or the file might be protected from read or write. The /usr/lib/uucp/USERFILE file lists valid path name prefixes. Some remote systems give this message if your system is not known to their system's L.sys file.

ACU LINE CLOSE (fail)
: The close on the ACU failed; this rarely occurs.

BAD READ ( )
: The login sequence did not get what it expected. Check L.sys to make sure the login sequence is correct. Use the `-x4` debug option of `uucico` or `send` to see what the other system is really saying.

CANNOT CALL (*system status*)
: An STST.*system* file exists and contains the status of the last call. If the status is not TALKING or CONVERSATION, you can remove this file. Status is left when a system cannot be reached. This prevents calls from being placed too often.

CANNOT DETERMINE (*system name*)
: All heuristics for learning the local system's name failed.

CAN'T LOCK (*resource name*)
: The specified resource could not be locked. If this happens repeatedly, *carefully* remove a lock file by hand.

CAN'T OPEN (*device or file name*)
: The device or file enclosed in parentheses was not available for opening when it should have been. If this happens repeatedly, be sure the file or device exists and that permissions on the file or parent directory are set correctly.

CAUGHT (*signal name*)
The specified signal hit UUCP. After cleanup, UUCP exited.

COPY (failed)
The file copy did not complete. It will be retried later.

DEBUG (enabled)
Self-explanatory.

DENIED (*can't open*)
UUCP tried to create a temporary file in a TM. directory in /usr/spool/uucp but failed. When this happens, it means the file system is full, so the directory is also full.

DONE (*work here*)
Someone requested UUCP to copy a file from the home system to the home system. It did so without dialing anywhere.

FAILED (ACU READ)
After successfully opening a modem for input/output, a read returned unsuccessfully. This happens when the modem hangs up or someone disconnects the cable.

FAILED (can't create TM)
Cannot create temporary file for transfer.

FAILED (can't read DATA)
A file was to be sent via `uucico`, but the file could not be read after all the setuid mechanisms. This happens when permissions in /usr/spool/uucp are incorrect or `uucp`, `uucico`, and other UUCP programs are not in agreement about the correct UID.

FAILED (connection refused)
An attempted UUCP connection over Ethernet failed because the remote system does not have the UUCP service defined in the /etc/inetd.conf database.

FAILED (dialup ACU write)
Bad return status from write to auto-dialer.

FAILED (dialup line open)
Auto-dialer did not return connection established. Possible reasons: no dial tone (the modem might be in talk mode), line busy, no answer (system down, wrong number).

FAILED (login)

The login sequence was unable to succeed during login at the remote site. If this happens repeatedly, use the `-x4` debug option of `uucico` or `send` to see what the remote site is really sending (and possibly expecting).

FAILED (conversation complete)
The call has been abnormally terminated.

FAILED (imsg 1) & FAILED (imsg 2)
The `uucico read` routines returned an EOF when least expected. This rarely occurs.

FAILED (to call system)
No connection to the call system could be made.

FAILED (startup)
No common protocol could be found.

HANDSHAKE FAILED ( )
The two hosts could not agree on whether the connection should proceed, or one host explicitly said no. This happens when more than one call is in progress between the two hosts or when someone has a LCK file set up for the host dialing in.

LOCKED (*system*)
A LCK.*system* file exists in /usr/spool/uucp indicating that a call is in progress to this system. If `uucp` died without removing this file or the system crashed and /etc/rc did not remove it, you can remove it. Do not remove it unless you are certain that `uucp` and other dialer-type programs are not running.

LOST LINE (login)
A read error occurred during login. This could be caused by the loss of the line when the remote system hangs up or by lack of response from the remote system.

MAIL FAIL (*username*)
Mail to someone failed and was returned to sender.

NO CALL (*max # of recalls*)
`uucico` gave up trying to call a site after too many failures. Remove STST.*sitename* file to try again.

NO CALL (retry time not reached)
No call was attempted because STST.*system* shows the last attempt failed and the retry time has not been reached. If you

are certain that the conditions causing the failure have been rectified, remove STST.*system* and try again.

NO (*device name*)
Cannot find L-devices device needed to make call (or all devices are currently locked).

NO WORK (*site name*)
Nothing is available to send to the listed site.

OK (conversation complete)
The call completed normally.

OK (startup)
A protocol has been agreed upon.

PATIENCE ( )
`uucico` is rereading a message.

PERMISSION (denied)
Cannot access remote site.

PREVIOUS (bad sequence)
Sequencing information between the two systems is wrong.

QUE'D (*filename*)
A file is queued to be transmitted.

REQUEST (*filename*)
A file is to be transmitted.

REQUESTED (*filename*)
A file is to be received.

REQUIRED (callback)
For security reasons, your system must call back the other system. This is a standard informative message.

SUCCEEDED (call to system)
A call has been placed, although login is not complete.

TABLE OVERFLOW ( )
`uucico` tried to open too many files. This rarely occurs.

TIMEOUT (dialup DN write)
Unable to write to auto-dialer. Check to make sure that the modem is asserting the DCD signal.

TIMEOUT ( )
An alarm expired and it is too late now. This can happen when one or both systems have high loads. If that is the case, try again when the loads are lower.

WRONG TIME TO CALL (*system name*)
Calls to this system can be made only during certain times (listed in /usr/lib/uucp/L.sys). Generally, `cron`, rather than user demand, starts long distance calls.

XQT QUE'D ( )
A remote execution has been queued for transmission.

XUUCP DENIED ( )
It is not legal to execute the specified program remotely. Fix the /usr/lib/uucp/L.cmds file on the remote system if you think you should be able to execute this command.

Example

The following example shows the format of the entries:

```
uucp ihnp4 (10/20-16:19-4619) OK \
(conversation complete)

INuucp smu (10/20-16:20-4794) SUCCEEDED \
(call to smu)

INuucp smu (10/20-16:21-4794) OK
(startup)
```

Programs

`uucico, uuxqt`

Caveats

The LOGFILE grows without boundaries at a site using UUCP. To save space, regularly discard all but the last few thousand lines.

/usr/spool/uucp/SYSLOG
UUCP file transfer log

Format

%s %s (%d/%d-%d:%d) (%d) %s data %d bytes %d secs

Description

The /usr/spool/uucp/SYSLOG file logs completed UUCP file transfers, listing the following:

- Name of user (usually a daemon) requesting the transfer
- System name
- Date and time
- Time in seconds since 1970
- Sent or received status of file
- Number of bytes
- Time required to complete the transfer

Example

```
daemon mazama (10/1-0:51) (496993902) sent \ data
671 bytes 5 secs

uucp allegra (10/1-0:53) (496994048) \ received
data 204 bytes 4 secs

daemon mazama (10/2-0:51) (496993907) sent \ data
62 bytes 0 secs
```

Programs

Summary programs have not been released for this file.

Caveats

The sum of the transfer times is not the long distance call time. The long distance call time, which is higher because of setup and takedown overhead, is not included in the transfer times.

The time to send a file can be short (0 seconds) because the teletype output system is buffered.

/usr/lib/uucp/USERFILE
list of valid path name prefixes for UUCP

Format

%s,%s

Description

The USERFILE specifies which file systems are accessible to local users and to remote systems using UUCP by describing the pathname prefixes that are valid for login IDs and systems. The USERFILE can be used to specify "automatic callback."

Each line of the USERFILE describes one user-machine pair. A comma (but no space) separates the user name and machine name. If either the user or machine entry is omitted, all users or machines are valid for the pair. If both user and machine entries are omitted, the entry matches all entries not previously matched.

If included, the character "c" (separated by spaces from the user-machine pair and paths) specifies callback. Callback causes the current conversation to end quickly and the current slave to call back the current master.

The final set of fields on each line is a set of path name prefixes. For security reasons, one path name of the set must match that of each file before it is transferred.

Example

```
,convex /
, /usr/spool/uucppublic
```

This example is a prototype USERFILE. Substitute your site's name for convex. Anyone on convex can send files away from the machine; explicitly: % ***uucp*** *file remote!path*. Machines calling in can move files to and from /usr/spool/uucppublic.

Programs

`uucico`

Caveats

If you change USERFILE from what is shipped with your system, you open your machine to possible security violations.

/usr/adm/acct
execution accounting file

Format Binary

Description The acct system call makes entries in an accounting file (/usr/include/sys/acct.h) for each process that terminates. The format of the account record has changed for versions of ConvexOS greater than V3.0. The new record contains larger fields to provide more precision and/or greater range. The new and old account records as defined by the include file are as follows:

```
/*        $CHeader: acct.h 1.7 88/08/31 09:52:58 $*/
/*        Copyright 1984 Convex Computer Corp.*/
/*
* Accounting structure
*/
structacct
{
     char        ac_comm[10];       /* Accounting command name */
     struct timevalac_nutime;       /* Accounting user time */
     struct timevalac_nstime;       /* Accounting system time */
     struct timevalac_netime;       /* Accounting elapsed time */
     time_t      ac_btime;          /* Beginning time */
     u_short     ac_uid;            /* (real) user ID */
     u_short     ac_gid;            /* (real) group ID */
     long        ac_aid;            /* activity ID */
     long long   ac_kbsec;          /* memory usage integral */
     long        ac_io;             /* number of disk IO blocks */
     dev_t       ac_tty;            /* control typewriter */
     char        ac_flag;           /* Accounting flag */
     char        ac_unused[5];      /* future expansion */
     float       ac_avconcur;       /* Average concurrency level */
};

#define     AFORK        0001       /* has executed fork, but no exec */
#define     ASU          0002       /* used super-user privileges */
#define     ACOMPAT      0004       /* used compatibility mode */
#define     ACORE        0010       /* dumped core */
#define     AXSIG        0020       /* killed by a signal */
#define AFIXSCH          0040       /* used fixed scheduling */

#ifdef KERNEL
struct acct         acctbuf;
struct vnode        *acctp;
#endif
```

If the process calls `execve`, the first 10 characters of the filename appear in `ac_comm`. The accounting flag contains bits that show whether `execve` was accomplished and whether the process ever had superuser privileges.

Programs

`sa`

Caveats

The /usr/adm/acct file grows without boundary unless you use `sa` to compress it. Use `cron` to run the accounting scripts to compress the file.

See Also

lastcomm(1), acct(2), execve(2), sumscripts(8)

/etc/activities
activity list

Format

%s%d

Description

The /etc/activities file is a list of activities that defines correspondences between activity names and activity IDs. For each activity, a single line contains the activity name, followed by a colon and the activity ID. To save time, sort the activity list, placing activities most often referenced at the beginning of the file.

Activity names cannot contain colons, and it is recommended that they not contain spaces (so as not to confuse `awk` scripts).

On login, activity ID is set to 0 (zero). For this reason, the activity with ID 0 should be a catch-all activity and should have an entry to this effect in the activities list. Typical names for this activity are "overhead" and "miscellaneous." An activity ID must fit in 32 bits and must fall in the range -2,147,483,648 to 2,147,483,647. There is a one-to-one mapping between activity names and activity IDs, so each activity name is associated with one activity ID and vice versa.

When used with the activity ID offset in CXbatch, activity IDs provide a means of keeping track of batch jobs. When a process is executed from a batch queue, the activity ID of the process is set to the activity ID of the user who submitted the job plus the number specified (if any) by the queue's activity ID offset. A suggested system is to number all activity IDs in increments of 10. Each batch queue would then have an activity ID offset from 1 to 9, with 0 reserved for jobs that were not submitted using the batch queues.

Example

```
misc:0
vctest:100
ostest:110
integration:500
tooldev:520
```

See Also

bill(1), idtoname(1), setaid(2), acct(5), actwho(5), qmgr(8)

/etc/actwho
group-activity access control file

Format

%s%s%s

Description

The actwho file is used by the `bill` command to determine who can bill to which group-activity combination. Each entry in the actwho file is of the form

group [.[*activity*]]: *users*

group A group listed in the /etc/group file.

activity An activity listed in the /etc/activities file. In the absence of an activity field, the activity with activity ID 0 is assumed. This activity is usually named "overhead" or "miscellaneous."

users Comma-separated list of user names allowed to bill to the group-activity combination given in the initial part of the entry.

group, *activity*, or *users* can consist of the single character * (asterisk), which stands for "all." *group* or *activity* can contain the ? character, which matches any character in that character position. With creative naming of activities, this capability can be used to allow certain users to bill to classes of activities. For example, suppose all activity names related to documentation were of the form "3820-*xxx*-7589", where *x* is some character that varies with the activity. Then an entry in /etc/actwho such as "*.3820-???-7589:user,..." would list users allowed to bill to documentation.

To reduce search time, /etc/actwho should be sorted, with group-activity combinations referenced most often placed at the beginning of the file. Use the `edactwho` utility to edit /etc/actwho; this utility assures correct syntax and coordinates file locking with the `bill` command.

Example

In the following example, users malone, erving, and shapira can bill to the group-activity combination of group nba and any activity.

```
nba.*:malone, erving, shapira
```

In the second example, all users can bill to the group-activity combination of group `sdev` and any activity starting with "017-" and ending in any four characters.

```
sdev.017????:*
```

See Also

bill(1), getacwent(3), activities(5), group(5), edactwho(8)

/usr/lib/aliases

`sendmail` aliases file

Format

%s

Description

This file describes user ID aliases used by `/usr/lib/sendmail`. Each line is of the form

name: name_1, name_2, name_3,...

where *name* is the alias for the list of *name_n*. Lines beginning with white space are continuation lines. Lines beginning with # are comments.

Aliasing occurs only on local names. Loops cannot occur, because messages are not sent to a person more than once. After aliasing is done, messages are forwarded to the list of users defined in each recipient's .forward file.

Example

The following example shows three aliases:

```
joans: jones
smithnotes:
 "|/usr/spool/notes/.utilities/nfmail
 smithnotes"
forum: garza, esmark, horwitz, chiarell,
 kleinfel, riley, keiko
```

The first alias corrects people who misspell user jones's name. The second forwards mail sent to "smithnotes" to the smithnotes notesfile. The third is a mailing list alias that forwards mail to many users.

Programs

`sendmail`

Bugs

Because of restrictions in `dbm`, an alias cannot contain more than aproximately 1000 bytes of information. For longer aliases, use an include file. You can also use a dummy name as the last name in the alias; this method is called "chaining." The dummy name is a continuation alias.

See Also

newaliases(1), dbm(3X)

/usr/adm/bill-acct
bill accounting file

Format Binary data

Description The /usr/adm/bill-acct file is a record of successfully executed `bill` commands. The bill.h include file defines the structure which is written twice each time a user changes billing accounts from a login shell. The first record corresponds to the group-activity combination that is terminating; the second corresponds to the new combination. The bill.h include file is shown in below:

```
/*  $CHeader: bill.h 0.0 86/01/28 18:37:17 $  */
/*  Copyright 1984 Convex Computer Corp.  */
struct billlog {
     short       bi_uid;            /* user id */
     short       bi_gid;            /* new/old group id */
     int         bi_aid;            /* new/old activity id */
     long        bi_time;           /* time as returned by time() */
     char        bi_tty[8];         /* tty line, as in <sys/acct.h> */
     unsigned short bi_flags;       /* see BI_xx #defines */
     char        bi_unused[4];      /* reserved for future use */
};

#define  BILLLOG  "/usr/adm/bill-acct"  /* log file */
```

See Also bill(1), utmp(5), connecttime(8), sumscripts(8)

/usr/lib/crontab
schedule of programs run by the `cron` utility

Format

%d %d %d %d %d %s

Description

The crontab file consists of lines of six fields each, separated by spaces or tabs. Each line lists a command and a time to execute the command. The first five fields are integer patterns specifying the following:

1. minute (0-59)
2. hour (0-23)
3. day of month (1-31)
4. month of year (1-12)
5. day of week (1-7, 1 = Monday)

Each pattern can contain one of the following:

- A number in the range listed
- Two numbers separated by a minus sign for an inclusive range
- List of numbers separated by commas for any of the numbers
- An asterisk (*) for all legal values

The sixth field is a string executed by the Bourne shell at the specified times. The shell translates percent characters in this field as newline characters. Only the first line (up to a % or end of line) of the command field is executed by the shell. The other lines (or whatever characters and lines following the %) are made available to the command as standard input. If a user forgets to redirect standard output or standard error, `cron` will mail the results of the command to the user.

The `cron` program examines /usr/lib/crontab every minute.

Example

In the following example, `cron` invokes a command daily at 5:00 a.m. The command examines files in /usr/preserve and deletes files more than one week old. Every hour (at 35 minutes past the hour), `cron` feeds the `runbuglist` script to a Bourne shell.

```
00 05 * * * find /usr/preserve -mtime +7 -a \ -
exec rm -f {}

35 * * * * setgid \

< /usr/local/lib/buglist/runbuglist
```

Programs `cron`

Caveats Scripts invoked by `cron` must include full path names for programs except those located in /bin and /usr/bin (unless you modify /.cronrc).

The search path for programs invoked by `cron` is usually more restricted than the search path invoked by most users; for instance, it often lacks /usr/local/bin and /usr/convex.

Format

Same as termcap file

(Refer to the termcap(3X) man page, which contains a detailed explanation of termcap-format files, while reading this /etc/disktab description)

Description

Each entry in /etc/disktab describes the partition layout for one type of disk.

Capabilities

Refer to termcap(5) for a description of the file layout. Table 36 shows /disktab types and descriptions.

Table 36 /disktab description table

Name	Type/description
ty	Type of disk
se	Number of bytes per sector
ns	Number of sectors per track
nt	Number of tracks per cylinder
nc	Number of cylinders per disk
rm	Disk speed (revolutions per minute)
p[a-h]	Partition sizes (sectors)
b[a-h]	Partition block sizes (bytes)
f[a-h]	Partition fragment sizes (bytes)

Example

```
dkd-001|dkd001|DKD-001|DKD001|eagle|Eagle| \
Fujitsu Eagle (45 sectors):
:ty=winchester:se#512:ns#45:nt#20: \
   nc#842:rm#3900 :pa#37800:ba#8192:fa#1024:
   :pb#151200:bb#8192:fb#1024:
   :pc#756000:bc#65536:fc#8192:
   :pd#37800:bd#8192:fd#1024:
   :pe#227700:be#8192:fe#1024:
   :pf#75600:bf#8192:ff#1024:
   :pg#341100:bg#8192:fg#1024:
   :ph#225900:bh#8192:fh#1024:
```

The example specifies data for a disk with the following names:

- `dkd-001`
- `dkd001`
- `DKD-001`
- `DKD001`
- `eagle`
- `Eagle`
- `Fujitsu Eagle (45 sectors)`

`Fujitsu Eagle (45 sectors)` is included as a descriptor. The disk type is `winchester`; the other type option is `removable`. The disk has 45 sectors per track and 20 tracks per cylinder for each of 842 cylinders. The disk spins at 3900 rpm. The a partition is 37,800 sectors (19,353,600 bytes) long. Major blocks are 8,192 bytes long; minor blocks are 1,024 bytes long. The example also specifies 7 other partitions.

Programs

The `newfs` and `newst` utilities determine physical characteristics for given disks by looking in /etc/disktab.

Caveats

Choose the layout of file systems carefully to avoid problems mounting file systems that share cylinders.

Do not change /etc/disktab except when adding a new style of disk to the system.

/usr/adm/diskuse/*
disk summaries for each user

Format

Each user has a file with a one-line description, for example:

%d %s

Description

Each file in the /usr/adm/diskuse directory corresponds to a user ID on the system and contains a series of lines. Each line in the file contains:

- Use in blocks
- Directory for which use is specified
- Date at which use is specified

Certain summary programs use this information to inform users of their excessive disk use.

Example

The following entry is from the file /usr/adm/diskuse/smith:

```
181 /mnt/smith  Fri Mar 18 05:26:47 CDT 1988
```

```
The /mnt/smith directory had 181 kilobytes on
Friday, March 18, 1988.
```

Programs

The following local programs can create and process data from the /usr/adm/diskuse files:

- `diskspace`—calculate usage
- `diskmail`—inform users of their disk use
- `quotaon`—turn file system quotas on
- `quotaoff`—turn file system quotas off
- `quotacheck`—file system quota consistency checker
- `repquota`—summarize quotas for a file system
- `edquota`—edit user quotas
- `quota`—display disk use and limits, manipulate disk quotas

/usr/adm/failure_log
log of file access failures

Format

Binary

Description

The /usr/adm/failure_log file is a log of all file accesses that failed because of insufficient permissions. To print the formatted log file to the screen, enter:

```
% /usr/adm/faillogpr /usr/adm/failure_log
```

Example

Each record in the file is printed on one line in the following format:

```
Tue Apr 14 16:08:29 CDT 1987 carson(rivers) RW
EPERM csh /etc/rc
```

`Tue`	Day of failed access
`Apr`	Month of failed access
`16:08:29`	Time central daylight time) of failed access
`carson`	Real user ID of accessor
`rivers`	Effective user ID of accessor (if different from real user ID)
`EPERM`	Error code
`RW`	Access code (read, write, execute) of the file
`csh`	Command being executed
`/etc/rc`	File user tried to access

Programs

`faillogpr`

Bugs

The failure log does not contain enough information to uniquely identify a file. For example, if the file in question is deleted after the failed access occurs, it is impossible to determine where that file was in the directory structure.

See Also

faillog(2), faillogon(8), faillogpr(8)

/etc/fstab
static information about file systems

Format

%s %s %s %s %d %d

Description

Each single-line entry in the ASCII /etc/fstab file describes the following fields:

1. Block (not raw) physical device described by the entry. For NFS partitions, this field is `machine:` / *full_path_name* .
2. Logical name upon which the device is normally mounted
3. Type of partition entry for this device:

`4.2`	Local disk
`nfs`	Network File System
`swap`	Swap partition
`ignore`	Comment line

4. Type of access for this device. The following options are valid on 4.2 and NFS file systems:

`rw`	Read/write (default)
`ro`	Read only
`suid`	Setuid execution allowed (default)
`nosuid`	Setuid not allowed
`quota`	Usage limits enforced
`noquota`	Usage limits not enforced (default)
`hide`	Ignore this entry during a mount -a command

The following options are available only on NFS systems:

`bg`	If first attempt fails, retry in the background
`fg`	Retry in foreground
`retry=`*n*	Set number of failure retries to *n*
`rsize=`*n*	Set read buffer size to *n* bytes
`wsize=`*n*	Set write buffer size to *n* bytes
`timeo=`*n*	Set NFS timeout to *n* tenths of a second
`retrans=`*n*	Set number of NFS retransmissions to *n*
`port=`*n*	Set server IP port number to *n*
`soft`	Return error if server does not respond

`hard` Retry request until server responds

5. Obsolete entry sometimes used to specify the dump frequency (in days)
6. `fsck` pass number for parallel dumps (and for running `fsck`)

The ASCII file contains an entry for each logical file system. Blanks or tabs separate the fields on each line. Comment lines start with #. Add an entry to this file each time you add a disk partition to the system. No program writes to /etc/fstab.

Example

```
/dev/da0d /mnt 4.2 rw 1 3
```

In this example, `mount -a` puts the /mnt file system on partition /dev/da0d for read and write access. No program currently examines the fourth field. The `preen` disk-checking program ignores the final "3," but `fsck` checks this file system concurrently with all other file systems with "3" as their pass number.

Programs

`df, fsck, preen, swapon, mount, umount`

Caveats

Mounting overlapping file systems causes loss of data on both partitions.

To access records from the /etc/fstab file, use `getmntent`.

Order records in /etc/fstab so `mount` and `umount` can process them sequentially. For example, if /master and /master/slave are partitions, you must include the /master entry in fstab before the entry for /master/slave.

Disk partitions specified as `swap`, `ignore`, and `nfs` do not have meaningful dump frequencies or pass numbers and are not checked by `fsck`.

You can include information on striped disk partitions in /etc/fstab.

See Also

/usr/include/fstab.h, stripecap(5), fstab(5)

/etc/gettytab
terminal configuration file

Format

Same as termcap

(Refer to the termcap(3X) man page, which contains a detailed explanation of termcap-format files, while reading this /etc/gettytab description)

Description

The gettytab file is a termcap-style file that describes terminal lines. The initial terminal login process `getty` reads the gettytab file each time it starts, allowing dynamic reconfiguration of terminal characteristics. Each entry in the database describes a class of terminals.

Capabilities

Refer to termcap(5) for a description of the file layout.

Table 37 lists types, defaults, and descriptions of entries in the terminal confiugration file. The `Default` column lists defaults obtained if no entry is specified and the special default table has no entry.

Table 37 Terminal configuration file

Name	Type	Default	Description
ap	bool	false	Terminal uses any parity
bd	num	0	Backspace delay
bk	str	0377	Alternate end of line character (input break)
cb	bool	false	Use CRT backspace mode
cd	num	0	Carriage-return delay
ce	bool	false	Use CRT erase algorithm
ck	bool	false	Use CRT kill algorithm
cl	str	NULL	Screen clear sequence
co	bool	false	Console add n after login prompt
ds	str	^Y	Delayed suspend character
ec	bool	false	Leave echo off
ep	bool	false	Terminal uses even parity
er	str	^?	Erase character

Table 37 Terminal configuration file (continued)

Name	Type	Default	Description
et	str	^D	End-of-file (EOF) character
ev	str	NULL	Initial environment
f0	num	unused	tty mode flags to write messages
f1	num	unused	tty mode flags to read login name
f2	num	unused	tty mode flags to leave terminal as
fd	num	0	Form-feed (vertical motion) delay
fl	str	^O	Output flush character
hc	bool	false	Do not hang up line on last close
he	str	NULL	Host name editing string
hn	str	host name	Host name
ht	bool	false	Terminal has real tabs
ig	bool	false	Ignore garbage characters in login name
im	str	NULL	Initial (banner) message
in	str	^C	Interrupt character
is	num	unused	Input speed
kl	str	^U	Kill character
lc	bool	false	Terminal has lowercase
lm	str	login:	Login prompt
ln	str	^V	"Literal next" character
lo	str	/bin/login	Program to execute when name obtained
nd	num	0	Newline (line-feed) delay
nl	bool	false	Terminal has (or might have) a newline character
nx	str	default	Next table (for auto-speed selection)
op	bool	false	Terminal uses odd parity
os	num	unused	Output speed
pc	str	0	Pad character

Table 37 Terminal configuration file (continued)

Name	Type	Default	Description
pe	bool	false	Use printer (hard copy) erase algorithm
pf	num	0	Delay between first prompt and following flush
ps	bool	false	Line connected to a MICOM port selector
qu	str	^	Quit character
rp	str	^R	Line retype character
rw	bool	false	Do not use raw for input, use cbreak
sp	num	unused	Line speed (input and output)
su	str	^Z	Suspend character
tc	str	none	Table continuation
to	num	0T	time-out (seconds)
tt	str	NULL	Terminal type (for environment)
ub	bool	false	Do unbuffered output (of prompts, etc.)
uc	bool	false	Terminal is known uppercase only
we	str	^W	Word erase character
xc	bool	false	Do not echo control characters as ^X
xf	str	^S	XOFF (stop output) character
xn	str	^Q	XON (start output) character

Example

```
default: \
   :ap:fd#1000:im=\r\n\r\nConvexOS, Release V9.0
(%h)\r\n\r\r\n\r:sp#1200:
2|std.9600|9600-baud:sp#9600:
```

`ap` Terminal uses any parity (default entry).

`fd` Form-feed delay is a full second.

`im` Initial banner message substitutes the name of the system for %h.

`sp` Default speed is 1200 baud.

As this example shows, only a few of the available options are typically used.

The "2" entry in `sp` changes nothing but the terminal speed. Additional entries (3, 4, and so on) also change the baud rate.

Caveats

If no line speed is specified, the line speed in use when `getty` is entered remains. Specifying an input or output speed overrides line speed for the stated direction only. CONVEX does not support split baud rates.

Terminal modes are derived from the Boolean flags specified (`f0`, `f1`, `f2`). Use these modes for message output, login name input, and to be sure no one changes tty modes. If the derivation should prove inadequate, any (or all) of these three can be overridden with the `f0`, `f1`, or `f2` numeric specification. These specifications identify (usually in octal, with a leading 0) the exact values of the flags. Local (new tty) flags are set in the top 16 bits of this 32-bit value.

When it receives a null character, `getty` restarts, using the table pointed to by `nx`. Null characters are presumed to show a line break. If no `nx` entry exists, `getty` reuses its original table.

You can precede the `cl` screen clear string with a (decimal) number indicating the number of milliseconds of delay required to clear the screen (termcap style). You can simulate this delay repeatedly using the pad character (`pc`).

To print the host name, include the character sequence `%h` in the initial message (`im`) and the login message (`lm`). `%%` prints a single % character. The host name is normally obtained from the system, but you can use the `hn` table entry to set it. You can edit the host name with `he`.

The `he` string is a sequence of characters—each character that is neither @ nor # is copied into the final host name. A @ in the `he` string causes one character from the real host name to be copied to the final host name. A # in the `he` string causes the next character of the real host name to be skipped. Surplus @ and # characters are ignored.

When `getty` executes the login process given in the `lo` string (usually /bin/login), it sets the environment to include the terminal type, as shown by the `tt` string. The `ev` string can be used to enter additional data into the environment. It is a list of comma-separated strings, with each string having the form *name=value*.

If you specify a nonzero time-out with `to`, `getty` exits within the given number of seconds, having received a login ID and passed control to `login`, or having received an alarm signal and exited. This is useful for hanging up dial-in lines.

Output from `getty` is even parity unless `op` is specified. Specify `op` with `ap` to allow any parity on input, but generate odd parity output. This only applies while `getty` is being run; terminal driver limitations prevent a more complete implementation. `getty` does not check parity of input characters in RAW mode.

Bugs

Some administrators change the default special characters, so it is wise to always specify (at least) the erase, kill, and interrupt characters in the default table. The characters # or ^H in a login name are treated as erase characters; @ is treated as a kill character.

The delay implementation is not flexible, and some of the delay algorithms are not used. The terminal driver should support useful delay settings.

Because `login` resets the environment, there is no point setting it in gettytab.

Programs

`getty`

See Also

termcap(5)

/etc/group
group file

Format

%s:%s:%d

Description

This ASCII file contains the following information for each group:

- Group name
- Encrypted password (OBSOLETE)
- Numerical group ID
- Comma-separated list of users allowed in the group

Each line describes one group; fields are separated by colons.

The file maps group names (the first field) to numerical group identification numbers (third field) and vice versa. The `ls` command, for example, uses this file.

Example

```
daemon:*:1:smith,jones
```

Group daemon (with * in the obsolete password field) is also known as group ID "1". Users smith and jones are in this group in addition to the group specified for them in /etc/passwd.

Programs

Several programs use the /etc/group file.

Caveats

Users become members of groups in the following ways:

- Having the group number listed in the /etc/passwd file
- Having their user name listed in the /etc/group file

See Also

/etc/passwd

bill(1), setgroups(2), initgroups(3X), crypt(3), passwd(1), passwd(5), getgrent(3), getgrgid(3), getgrname(3), setgrent(3), endgrent(3), getgroups(2)

Format

%s

Description

The hosts file contains information on known hosts on the DARPA Internet. For each host, a single line containing the following information must be present:

- Official host name
- Internet address (in 4-octet format)
- Aliases by which the host is known

Blanks or tabs separate items. Comments start with #; characters from # to the end of the line are not interpreted by routines searching the file. The /etc/hosts file is created from the official host database maintained at the Network Information Control Center (NIC). Make local changes to add unofficial aliases or unknown hosts (for example, your own local host name).

Network addresses are specified in the conventional "." notation using the `inet_addr` routine from the Internet address manipulation library, `inet`. Host names can contain any printable character except a field delimiter, newline, or comment character.

Example

The following example shows the local portion of the /etc/hosts database. The lines following it in /etc/hosts come from NIC distribution tables or from the CONVEX distribution.

```
# Berkeley Host Database

127.1localhost

# Convex 10Mb/s local Ethernet

100.0.1.0x82convexs-ex  convexs sw

100.0.0.1convexe-ex

100.0.0.6convex1-ex  convex1
```

The `127.1 localhost` line defines a host address (for this host) used for testing, loopbacks, and some daemons.

The local hosts (100.*) are members of the local Ethernet. The "100" prefix choice is arbitrary, but choices should be compatible with existing host numbers.

When an unknown name is requested, the `arp` name-resolution protocol broadcasts packets requesting identification.

Programs `ftp, rcp, rdist, rex, rlogin, rsh, sendmail`

Bugs Use a name server instead of a static file. A binary indexed file format should be available for fast access.

See Also gethostent(3N), rcmd(3X), arp(4)

CONVEX Network File System System Manager's Guide

/usr/adm/lpd-acct
printer accounting file

Format

%d%d%s%s%s

Description

Printer accounting is done by the print filters. There can be more than one printer accounting file, as determined by /etc/printcap. However, most ConvexOS filters use /usr/adm/lpd-acct. For consistency, these filters use the same log routine. Accounting data in this format can be summarized by `awk` scripts as described in sumscripts(8).

Example

```
#include <stdio.h>

log(acctfile, pages, userinfo, logname, host)

char *acctfile;

intpages;

char*userinfo;

char*logname;

char*host;

{
   FILE*fptr;
   /* missing from filter argument list */
   if (acctfile == NULL)
   return;
   if ((fptr = fopen (acctfile, "a")) == NULL)
   return;
   fprintf (fptr, "%10d %6d %s %8s %8s n",
   time (0), pages, userinfo, logname, host);
   fclose (fptr);

}
```

The arguments `acctfile`, `logname`, and `host` are generated as described by the "Berkeley 4.2BSD Line Printer Spooling Manual" in the *ConvexOS Tutorial Papers*. The `pages` argument is computed by the filter. In ConvexOS, if the `pu` capability (propagate user information) is present in a printer's /etc/printcap entry, `lpd` passes the command line arguments `-u userinfo` to the printer's filter(s). The `userinfo` portion can then be passed to the log routine.

See Also

bill(1), printcap(5), lpd(8), pac(8), sumscripts(8)

/etc/motd
the message of the day

Format

ASCII text

Description

The `login` program displays /etc/motd when a user who has not seen the message logs on. The file only displays one time for each user login.

Use /etc/motd to send messages of general interest to all users.

Example

The following message is an example of an /etc/motd file entry:

```
9/10/82: The system will be down all day \
Saturday for updates.
```

Programs

`login`

See Also

/etc/nologin

/etc/nologin
inhibit logins and print message

Format ASCII text

Description If an /etc/nologin file exists, `login` does the following:

- Denies login to everyone except root
- Prints contents of /etc/nologin file as the reason for login denial

Example The following message is an example /etc/nologin file:

```
System unavailable -- operator performing special
file backup operations. Try again at 17:00 hours.
```

Programs `login, shutdown`

Format %s:%s

Description The /etc/nurc file contains default constants for the nu utility. The nu utility adds new users to the system; nu uses the constants defined in /etc/nurc if no other values are provided and if an alternate default file is not specified in place of /etc/nurc.

Lines in the nurc file have the form:

tag: *value*

tag is the parameter to modify and *value* is the default constant for the parameter. The tags and their default values are shown in Table 38.

Table 38 /etc/nurc

Name	Default	Description
uid	(see below)	User ID
gid	EMPTY	Group ID
group	staff	Group name (used if no GID field)
directory	/mnt	Path for home directory
protection	0755	Home directory protection
shell	/bin/csh	Login shell
password	(see below)	New user's initial password
username	username	User's full name (for finger)
office	office	User's office (for finger)
extension	extension	User's work extension
homephone	homephone	User's home phone number
minwks	1	Minimum # weeks for password aging
maxwks	52	Maximum # of weeks for password aging
diskquota	6000,8000,1200,1500	blksoft, blkhard, inodesoft, inodehard quotas
skeleton	/usr/skel	Directory of files to copy to homedir

Table 38 /etc/nurc (continued)

Name	Default	Description
homedir	< *directory* >/< *login*>	Home directory to create for user
sgroup	"staff"	Parent share group of new user
shares	"100"	Number of shares assigned to new user
typed	OFF	Will user have password typing restrictions?
aged	OFF	Will user have password aging restrictions?
quota	OFF	Will home directory file system have quotas initialized?
nouidfile	OFF	Ignore the contents of /etc/uidcount?
newsgroup	OFF	Is this login a new share scheduling group?
notshared	OFF	Is this login a not-shared scheduling group?

For more information on these parameters, see the nu(8) man page.

Example

The following is an example nurc file:

```
directory: /mnt
protection: 0700
shell: /bin/csh
group: swtst
username:
typed
quota
diskquota: 3000, 4096, 1000, 1500
```

In the example, new user's directories are created in the /mnt directory, which is readable, writable, and executable only by the user (protection: 0700). The default group is `swtst`. Typing restrictions are activated (although password aging restrictions are not), and the home directory file system has quotas initialized.

See Also

chfn(1), finger(1), nurc(5), passwd(5), nu(8)

/etc/op.access

op (operator) access file

Format

%s %s [%s [%s]]; [%s [%s]]

Description

The op.access file describes the functions a user can perform by invoking the op utility, how these functions are performed, and who is allowed to perform them. Because op.access is used to grant command access to commands that require superuser privileges, this file should be owned by root with mode 0400 (read only).

Each entry in the op.access file describes a single function. Lines beginning with a pound sign (#) are comments. If a line begins with a tab or spaces, it is considered part of the previous line. The entries describe the following:

- Name or mnemonic of the operator function
- Command line (full path name) to execute to perform this function
- UID to set (root by default)
- GID to set (not changed, by default)
- Directory to `chdir` (not changed, by default)
- Root directory to set with `chroot` (not changed, by default)
- `umask` to set (022 by default)
- List of groups allowed to execute the function (none by default)
- List of users allowed to execute the function (superuser only, by default)
- Range of valid arguments for the command (any value per variable argument by default)
- Environment variable settings (none by default)

Entry fields in op.access are of the form

mnemonic command [*arg* [...]] ; [*option* [...]]

where

mnemonic Unique identifier for the operator function

command Full path name of command executed when the mnemonic is called with op

arg Literal or variable arguments needed by command

option Option or set of options restricting who can use command or how command is invoked. The format for option is

keyword = *value* [,...]

where

keyword is the name of the option; *value* is the value assigned to the option.

Example

```
# define the site defaults; we want the people in # 'operators' group to
be able to execute almost # everything; easier here than on every line.
DEFAULT          groups=operators
# filesystem backups
weekly           /etc/dump 0Gun $1; users=snow,white
                 $1=/,/usr,/mnt,/project
daily            /etc/dump 5Gun $1; users=snow,white
                 $1=/,/usr,/mnt,/project
#
# taking system down; $1 is good use of regular  # expressions
shutdown         /etc/shutdown -h $1 $2; $1=now,+[0-9]*,[0-9]*:[0-9]*
reboot           /etc/shutdown -r $1 $2; $1=now,+[0-9]*,[0-9]*:[0-9]*
#
# kill all batch procs so system can be restarted # (want to override
default groups setting given # at top, allowing none)
killbatch        /usr/convex/qmgr shutdown 0;
                 groups= users=bashful,happy
startbatch       /usr/lib/nqs/nqsdaemon;
#
# start disco daemon
disco            /etc/opbin/start_disco; uid=disco gid=proj
                 dir=/scratch
                 umask=027 groups=geo,disco users=snoopy,woodstck
                 $USER=disco $SHELL=/bin/shell
#
# let particular people mount and unmount the removable drive
rdsmount         /etc/mount $1 $2; groups=operators,swdev,disco dir=/
                 users=bob,steve $1=/dev/dd0.$2=/.*
rdsumount        /etc/umount $1; groups=operators,swdev,disco dir=/
                 users=bob,steve $1=/dev/dd0.
```

From this example, users with operator privileges could use the following commands:

```
% op weekly /mnt
```

```
% op reboot 17:30 "We have to fix our \ network."
% op disco
% op rdsmount /dev/dd0c /mnt/me/mystuff
% op mounted 3 8688
```

Programs op

Caveats For security reasons, this file is shipped as read-only by root (mode 0400); do not change these permissions.

See Also regex(3)

/etc/passwd
password file

Format

%s:%s:%d:%d:%s:%s:%s

Description

Each single-line entry in the ASCII /etc/passwd file maps a sign-on name to the following seven fields:

1. Name
2. Encrypted password
3. Numerical user ID number
4. Group ID number
5. Full user name, office, extension, and home phone number
6. User's home directory
7. User's default command interpreter (shell)

The ASCII file contains an entry for each user on the system. Colons separate the fields on each line. If the password field is empty, no password is required. A null command interpreter field defaults to the `/bin/sh` interpreter.

The `nu` program appends new users to the /etc/passwd file. The following programs modify the user information when necessary:

- `chsh` changes the user's shell.
- `chfn` changes the user's identification field.
- `passwd` changes the user's password.

Example

```
smith:2sadfNQDlxsMg:11:19:Rob \
&,22A,228,3823133:/mnt/smith:/bin/csh
```

In this example, Rob Smith has the sign-on name `smith` (first field). The second field is his encrypted password. His user identification number is `11`; his group ID is `19`. The fifth field (user identification) has four comma-separated subfields. His full name is `Rob Smith` (the `&` causes the user name to be substituted appropriately); his office is `22A` with office phone `228` and home phone `382-3133`. His home directory is /mnt/smith, and he uses the standard `C` shell command interpreter.

Programs

`ls, finger, passwd, nu, chfn, chsh, vipw, login`

Caveats

System users are generally permitted read privileges to /etc/passwd because the passwords are encrypted. Write permission should be reserved for the superuser. Read permission is necessary because programs use the /etc/passwd file to map user IDs to user names.

The password file can be edited with a text editor, but it is important to ensure that only one person at a time edits the file. The editing programs `chfn`, `chsh`, `passwd`, and `vipw` are preferred because they perform the required file locking. Be aware that writing out an ill-formatted passwd file can create problems with programs that use the ill-formatted lines and may introduce security holes.

See Also

/etc/group

bill(1), crypt(1), group(5), getpwent(3), getpwuid(3), getpwnam(3), setpwent(3), endpwent(3)

/etc/phones
remote host phone number database

Format

%s %s

Description

The file /etc/phones contains the system-wide private phone numbers for the `tip` program. This file is normally unreadable, and so may contain privileged information.

Each line contains a system name followed by a phone number. The system name is one of the names defined in the /etc/remote file; the phone number is constructed from [0123456789-=*%K]. The =, K, and * characters cause the auto-call units to pause and wait for a second dial tone (when going through an exchange). VADIC modems use the K character.

Only one phone number per line is permitted. If more than one line in the file contains the same system name, `tip` attempts to dial each one in turn until a connection is made.

Example

```
orpheus   9K7654321
```

Programs

`tip`

/etc/printcap
printer capability database

Format

Same as termcap.
(Refer to the termcap(3X) man page, which contains a detailed explanation of termcap-format files, while reading this /etc/disktab description)

Description

/etc/printcap is a termcap-style file that describes your system's set of line printers. The spooling system reads the printcap file each time a file is to be printed, so you can add and delete printers dynamically. Each entry in the file describes one printer.
Refer to the chapter, "Setting up the line printer system" for instructions on setting up an entry for a particular printer.

Capabilities

Refer to termcap(5) for a description of the file layout. Table 39 lists types, defaults, and descriptions of entries in the /etc/printcap file.

Table 39 /etc/printcap

Name	Type	Default	Description
af	str	NULL	Name of accounting file
br	num	none	If lp is a tty, set baud rate (ioctl)
cf	str	NULL	cifplot data filter
df	str	NULL	TEX data filter (DVI format)
fc	num	0	If lp is a tty, clear flag bits (sgtty.h)
ff	str	f	String to send for form feed
fo	bool	false	Print a form feed when device is open
fs	num	0	Similar to fc, but set flag bits
gf	str	NULL	Graph data filter (plot format)
ic	bool	false	Driver supports (nonstandard) ioctl to indent printout
if	str	NULL	Name of text filter that does accounting
lf	str	/dev/console	Error logging filename
lo	str	lock	Name of lock file

Table 39 /etc/printcap (continued)

Name	Type	Default	Description
lp	str	fI/dev/lp	Device name to open for output
mx	num	1000	Max file size (in BUFSIZ blocks), 0= no limit
nd	str	NULL	Next directory for list of queues (N/A)
nf	str	NULL	ditroff data filter (device-independent troff)
of	str	NULL	Name of output filtering program
pl	num	66	Page length (in lines)
pu	bool	false	Propagate user information to filters
pw	num	132	Page width (in characters)
px	num	0	Page width in pixels (horizontal)
py	num	0	Page length in pixels (vertical)
rf	str	NULL	Filter for printing FORTRAN style text files
rm	str	NULL	Machine name for remote printer
rp	str	lp	Remote printer name argument
rs	bool	false	Restrict remote users to those with local accts
rw	bool	false	Open printer device for reading/writing
sb	bool	false	Short banner (one line only)
sc	bool	false	Suppress multiple copies
sd	str	/usr/spool/lpd	Spool directory
sf	bool	false	Suppress form feeds
sh	bool	false	Suppress printing of burst page header
st	str	status	Status file name
tf	str	NULL	Troff data filter (cat phototypesetter)

The *ConvexOS Man Pages* describe these capabilities.

Example

```
fast|Fast Imagen serial printer:
     :af=/usr/adm/imagen: :br#19200:
     :df=/usr/local/lib/idvi:fc#0177777:ff= 00:fo=0:
     :fs#040:gf=/usr/local/lib/igraph:if=/usr/local/lib/ipf:
     :lf=/usr/spool/nipd/ilog:lo=lock:lp=/dev/imagen2:
     :mx=#10000:nf=/usr/local/lib/idimp:pl#60:pw#80:
     :px#2016:py#2624:rf=/usr/local/lib/ifort:rw:
     :sb=CONVEX Computer Corporation:sc:sd=/usr/spool/nipd2:
     :sh:st=/usr/spool/nipd2/pstatus:vf=/usr/local/lib/imp:
```

`fast`	Printer name stands for "fast Imagen printer".
`af`	Accounting information goes to the /usr/adm/imagen file.
`br`	Baud rate is 19200.
`df`	Text data filter `/usr/local/lib/idvi` preprocesses the data before it is printed when you use the `lpr` command and specify the TEX option.
`fc`	All sgtty.h style bits are cleared on startup.
`ff`	Form-feed strings are nulls.
`fo`	No form feed is sent when the device is opened.
`fs`	Flag bits 040 (octal) are set on startup.
`gf`	Graphics filter /usr/local/lib/igraph preprocesses the data before it is printed when you use the `plot` command.
`if`	Accounting data is processed by filter /usr/local/lib/ipf.
`lf`	Errors are logged to the /usr/spool/nipd/ilog file.
`lo`	Lock file is lock (default).
`lp`	Printer name is /dev/imagen2.
`mx`	Maximum file size is 10,000 blocks.
`nf`	`ditroff` output goes through the /usr/local/lib/idimp filter.
`pl`	Page length is 60 lines.
`pw`	Page width (line length) is 80 characters.
`px`	Page width (horizontal dimension) is 2016 pixels.
`py`	Page height (vertical dimension) is 2624 pixels.

`rf`	FORTRAN listings go through the /usr/local/lib/ifort filter.
`rw`	Line opens in read/write mode.
`sb`	Short banner is "CONVEX Computer Corporation".
`sc`	Multiple copies suppressed.
`sd`	Spool directory is /usr/spool/nipd2.
`sh`	Burst page headers suppressed.
`st`	Status messages appear in /usr/spool/nipd2/pstatus.
`vf`	Raster images go through the /usr/local/lib/imp filter.

Programs `lpc, lpd, pac, lpr, lpq, lprm`

Caveats Be sure to specify # or = carefully, depending on whether a parameter has a numerical or string value.

See Also lpd-acct(5), termcap(5), lpd(8)

The chapter, "Setting up the line printer system," in this manual.

/etc/pwrestrict
password restrictions file

Format

%s:%s:%s:%s:%d

Description

Each single-line entry in /etc/pwrestrict maps a sign-on name to the following fields:

- Name
- Encrypted password
- Password aging information
- Password type flag (Y or N)
- Numerical user identification number

The pwrestrict file contains an entry for each system user. Colons separate the fields on each line. If the aging information field is empty, password aging is not enforced.

The `nu` program appends new users to the pwrestrict file. `passwd` modifies the user's password and the password aging information.

Example

```
smith:ENEaEinahyIAo:1,2,Oct 13 1986:Y:469
```

In this example, the sign-on name is `smith` (first field). The second field is the encrypted password. The password age field denotes that the password was last changed on Oct 13 1986; the user cannot change the password for one week (from the given date) and is forced to change the password after Oct 27 1986 (2 weeks after the last-changed date). The fourth field is a flag (Y or N); if the flag is N, the password can be composed of any sequence of characters. If the flag is Y, the user's password must follow the guidelines for passwords described in `passwd`. The user identification for `smith` is `469` (the last field).

Programs

`passwd, pwage, nu, vipw, login`

Caveats

Users are permitted read privileges to /etc/pwrestrict because the passwords are encrypted. Reserve write permission for the superuser. Read permission is necessary for programs that use the /etc/pwrestrict file to map user IDs to user names.

The pwrestrict file can be edited with a text editor, but only one person should edit the file at a time. The editing programs `passwd`, `vipw`, and `nu` are preferred because they perform the required file locking. Writing an ill-formatted pwrestrict file can cause problems with programs that use the ill-formatted lines. If inconsistencies are found between the /etc/passwd file and the /etc/pwrestrict file, information from the password file is used; the restriction information is ignored.

See Also

/etc/passwd, /etc/group

crypt(3), group(5), getpwrestent(3), getpwrestuid(3), getpwrestnam(3), setpwrestent(3), endpwrestent(3), passwd(5)

/etc/rc.local
system-specific startup information

Format Shell script

Description The system startup routine processes the /etc/rc.local file after mounting the file systems but before starting the daemons. The rc.local file contains shell commands that perform the following tasks:

- Set the host name
- Set up networks
- Check quotas
- Save core dumps
- Start local daemons
- Remove temporary files after crashes

Example

```
exec >/dev/console 2>&1
PATH=/etc:/usr/etc:/bin:/usr/ucb:/usr/bin:/usr/c onvex
export PATH
/bin/hostname convex
/etc/ifconfig ex0 convex-ex up arp -trailers
echo -n 'check quotas: '
     /usr/etc/quotacheck -a
echo 'done.'
/usr/etc/quotaon -a
echo -n 'local daemons:'
if [ -f /etc/routed ]; then
     /etc/routed & echo -n ' routed'
fi
if [ -f /etc/rwhod ]; then
     /etc/rwhod & echo -n ' rwhod'
fi
if [ -f /usr/lib/nqs/nqsdaemon ]; then
     /usr/lib/nqs/nqsdaemon
     echo -n ' cxbatch
fi
echo '.'
rm -f /tmp/Emacs* /tmp/queue?/*
rm -f/usr/spool/notes/.locks/*
/usr/convex/spu -r /mnt/os/vmunix > /vmunix
/usr/convex/spu -r /mnt/errlog > /errlog.back
```

See Also

Descriptions of the daemons, editor temporary files, and administrative support programs.

/etc/remote
remote host description file

Format

Same as /etc/termcap.

Description

Systems known by `tip`, and their attributes, are stored in the termcap-style file /etc/remote. Each line in the file describes a remote host.

The first entry is the name of the host system; vertical bars separate multiple names for a single system.

The `tip` program and the `cu` interface to `tip` use entries named "tip*" and "cu*" for defaults as follows. When `tip` is invoked with only a phone number, it looks for an entry of the form "tip300", where "300" is the baud rate with which the connection is to be made. When you use the `cu` interface, it looks for entries of the form "cu300".

Capabilities

Capabilities are either strings (str), numbers (num), or Boolean flags (bool). A string capability is specified by *capability=value*, for example, `dv=/dev/harris`. A numeric capability is specified by *capability#value*, for example, `xa#99`. A Boolean capability is specified (as true) by listing the capability, as shown in Table 40.

Table 40 /etc/remote

Name	Type	Default	Description
`at`	str		Auto call unit type
`br`	num	300	Baud rate
`cm`	str		Initial connection message to be sent to remote host
`cu`	str	(dv)	Call unit if making a phone call
`di`	str		Disconnect message sent to host on user request
`du`	bool		Host is on a dialup line
`dv`	str		UNIX device(s) to open to establish a connection
`el`	str	NULL	Characters marking an end-of-line
`fs`	str	BUFSIZ	Frame size for transfers
`hd`	bool		The host uses half-duplex communication (do local echo)
`ie`	str	NULL	Input end-of-file (EOF) marks

Table 40 /etc/remote (continued)

Name	Type	Default	Description
oe	str	NULL	Output end-of-file (EOF) string
pa	str	even	Parity used when sending data to the host
pn	str		Telephone number(s) for the host
tc	(str)		Continue capability at string in named description

Example

```
UNIX-1200:\
        :dv=/dev/cua0:el=^D^U^C^S^Q^O@:du:\
        :at=ventel:ie=#$%:oe=^D:br#1200:
arpavax|ax:\
        :pn=7654321%:tc=UNIX-1200:
```

In this example, the `UNIX-1200` device (`dv`) uses /dev/cua0, which connects to a ventel modem (`at`) on a dialup (`du`) line. Any of the entire line (`el`) list characters (`^D`, `^U`, `^C`, `^S`, `^Q`, `^O`) cause the entire line and the character to be sent to the remote host. The connect program sends an output end-of-file (`oe`) string `^D` after transferring a file. The characters #, `$`, and `%` signify end-of-input-file when preceded by `\n` (shell prompts). The `arpavax` name specifies a phone number (`pn`) to dial and then the standard `UNIX-1200` characteristics.

Programs

`tip`

Caveats

If you use `dv` to refer to a terminal line, `tip` tries to open it for exclusive use.

The ~ (tilde) escapes are only recognized by `tip` after an `el` character or a carriage return.

You can set parity to even, odd, none, zero (always set bit 8 to zero), or 1 (always set bit 8 to 1).

If the phone number (`pn`) field contains an @ sign, `tip` searches the /etc/phones file for a list of telephone numbers.

Bugs

The `ie` facility for specifying input EOF does not work with all files.

See Also

phones(5)

/usr/adm/shutdownlog
list of voluntary system shutdowns

Format

ASCII

Description

Each line in the shutdownlog file notes the time your machine was shut down, by whom, and the reason (if specified in the `shutdown` invocation).

Example

```
19:15 Sat Oct 12, 1985. Halted.
```

```
13:25 Mon Oct 14, 1985. Shutdown: for 20 min (by convex!root)
```

```
13:25 Mon Oct 14, 1985. Halted.
```

In this example, the machine was down twice.

Programs

`shutdown`

Caveats

The shutdownlog file can become quite large, but it is important to keep.

/usr/adm/stat/*
system statistics

Format

Binary

Description

Each file in the /usr/adm/stat directory corresponds to a date. The statistics gathered by the `usr/etc/stat` program for that date reside in the corresponding stat file. The `usr/etc/stat` program awakens approximately every 30 seconds and samples system use, disk activity, load average, number of users signed on, etc., then writes the data to the current file in the /usr/adm/stat directory. The format of the date is *mm.dd.yy* (lower months and days are single digit). Use the `seestat` program to examine the contents of the stat files.

`cron` usually runs `seestat` early in the morning.

Programs

`seestat`

Caveats

The stat files use a lot of disk space; archive them monthly.

/etc/stripecap
striped disk partition description database

Format

Same as /etc/termcap.

Description

The /etc/stripecap file is a database that describes striped disk partitions. The `putst` and `newst` utilities use the stripecap.

Striped disk partitions are logical disk partitions spanning several physical disk partitions. Striped disk partitions exploit performance improvements made possible by the parallel operation of several disk arms. ConvexOS file systems can be mounted on striped disk partitions just as with conventional disk partitions.

Striped partitions are described in the /etc/stripecap file by a termcap-style descriptor. This descriptor contains information on the physical disk partitions that constitute the striped partition and on the layout of the logical sections of the stripes.

The `newst` and `putst` utilities assume each stripe partition has a name in the format `st`*X*, where *X* is a numerical digit corresponding to the minor device number of the stripe disk pseudodevice /dev/rst*X*.

Capabilities

Refer to termcap(5) for a description of the file layout. Table 41 lists the types and descriptions of entries in the /etc/stripecap file.

Table 41 /etc/stripecap

Name	Type	Description
np	num	Number of partitions constituting the stripe file system
M?	num	Major device number of partition ?
m?	num	Minor device number of partition ?
D?	num	Number of devices (partitions) in section ?
B?	num	"stripe blockhouses" (interleave factor) of section ?
S?	num	Number of blocks in section ? contributed by each partition

Example

To update the /etc/stripecap file, use the following command sequence:

```
% /etc/newst /dev/rst6 da4a dkd-001 da4h \
dkd-001 da4g dkd-001
```

The output is the following stripecap entry:

`st6: \`	Name of this stripe device.
`:np#3:\`	Number of partitions included (three).
`:M0#5:m0#38:\`	Device numbers of three partitions
`:M1#5:m1#39:\`	sorted from largest size to
`:M2#5:m2#32:\`	smallest size.
`:D0#3:B0#37800:S0#128:\`	First section uses all three partitions.
`:D1#2:B1#188100:S1#128:\`	Second section uses first two partitions.
`:D2#1:B2#115200:S2#128:\`	Third section uses first partition.

To continue entries onto multiple lines, specify \ as the last character of a line. You can include empty fields (adjacent colons) for readability (for example, between the last field on a line and the first field on the next).

All fields have names that are two-character codes. For most field names, the second character is uniquely chosen from the set [0-9a-zA-Z] in an ascending sequence. For example, the minor device numbers of the first five disk partitions are `m0`, `m1`, `m2`, `m3`, and `m4`.

The name of this stripe device is `st6`, which means that /dev/st6 is the name on which to mount the file system. It includes three partitions on device number pairs 5,38 and 5,39 and 5,32. The order of these is important: 5,38 must be the largest partition; 5,32 must be the smallest. The stripe system interleaves all three partitions until the third one's space is exhausted. It then uses the first two partitions, and finally only the first partition (assuming unequal sizes of partitions, which is not necessarily the case).

In this example, the first section uses parts of all three partitions (for the first 37,800 sectors). The `S?` parameter directs the striping system to take the first 128 sectors from 5,38; the next 128 sectors from 5,39; then the next 128 sectors from 5,32. The fourth set of 128 sectors comes from 5,38 (again), and the cycle continues until 37,800 sectors have been used from each partition.

The second section repeats the sequence by using 128-sector chunks from only the first and second partitions. The third section uses only the remaining space on the first partition.

Programs `newst, putst, st`

/etc/syslog.conf
syslog configuration file

Format

facility.level; send_message_here

Description

Entries in the /etc/syslog.conf file determine the types of messages to be logged by `syslogd` and where to log the messages. Each entry has a selector that determines the message priorities and an action. The facility and level must be separated by a dot and each entry must end with a semicolon. The facilities available are as follows:

`kern`	Messages generated by the kernel (unused by CONVEX, but recognized in case of forwarding)
`user`	Messages generated by user processes
`mail`	Messages generated by the mail system
`daemon`	Messages generated by system daemons
`auth`	Messages generated by the authorization system
`lpr`	Messages generated by the line printer spooling system
`syslog`	Messages generated by syslog
`news`	Messages generated by news
`uucp`	Messages generated by UUCP
`covue`	Messages generated by COVUE
`tape`	Messages generated by the tape system
`batch`	Messages generated by the CXbatch product
`cron`	Messages generated by the `cron` utility
`local`*n*	Reserved for local definition. *n* can be any number between 1 and 7.

More than one facility can be selected, separated by commas. Use an asterisk (*) to indicate all facilities.

The levels available are as follows:

`emerg`	Panic condition; normally to all users
`alert`	Condition to be corrected immediately; for example, corrupt database
`crit`	Critical conditions; for example, hard-device errors

`err`	Errors
`warning`	Warning messages
`notice`	Conditions that are not errors but may require special handling
`info`	Informational messages
`debug`	Information useful for debugging a program

The following format alternatives can be used for the message destination (that is, where the message is to be logged):

- Absolute path name
- Host name preceded by @ sign
- List of user names, separated by commas (users receive messages if they are logged on)
- Asterisk (*) to send message to all logged-in users

Caveats

If you modify the default syslog.conf file, you must kill and restart `syslogd` after making the changes.

Example

```
# critical and authorization stuff
# gets wide notice
#
*.warn;user,kern.none/usr/adm/log/critical
*.warn;kern,user.none;auth.info/dev/console
*.warn;kern,auth.info;local7.none@convex
kern.debug              /usr/adm/log/kernlog
auth.debug              /usr/adm/log/authlog
daemon.debug            /usr/adm/log/daemonlog
lpr.debug               /usr/adm/log/lpd-errs
mail.debug              /usr/spool/mqueue/syslog
syslog.debug            /usr/adm/log/syslog
user.debug              /usr/adm/log/messages
```

Programs

`syslogd, logger`

Format

```
char PC;
char *BC;
char *UP;
short ospeed;

tgetent(bp,name);
char *bp, *name;

tgetnum(id)
char *id;

tgetflag(id)
char *id:

char *
tgetstr(id,area)
char *id, **area;

char *
tgoto(cm,destcol,destline)
char *cm;

tputs(cp,affcnt,outc)
register char *cp;
int affcnt;
int (*outc)();
```

Description

The functions listed in the "Format" section extract and use capabilities from the terminal capability database, termcap.

See Also

stripecap

/usr/adm/tp-acct
tape allocation accounting file

Format

%s%d[%s]%d%d%d%s

Description

The /usr/adm/tp-acct file contains a record of tape allocations and deallocations performed by `tpmount` and `tpunmount`. Each successful allocation or deallocation writes a line to /usr/adm/tp-acct containing the following fields:

event	`allocated` or `deallocated`
time	Time event occurred, as returned by `time`
device	Physical device allocated (present only if event is 'allocated')
uid	User ID of allocator at time of allocation
gid	Group ID of allocator at time of allocation
aid	Activity ID of allocator at time of allocation
drive	Tape drive name as given in the configuration database maintained by `tpconfig`

See Also

tpmount(1), tpconfig(5), sumscripts(8)

Format

%s

Description

`init` reads the ttys file to determine the terminal files for which to create `getty` processes that enable users to login. Each file is on a separate line in the ttys file. Fields are separated by tabs or spaces; if a field contains more than one word, the words must be enclosed in double quotation marks. Blank lines and comments can appear anywhere in the file; comments are delimited by the # symbol and a new line. Unspecified fields default to null.

The first field is the entry in /dev for the terminal.

The second field is the command, usually `getty`, to execute for the line. `getty` performs baud-rate recognition, reads the login name, and calls login, among other tasks. The second field can be a command, for example, to start up a window system terminal emulator or another daemon process; `getty` is the most commonly used, but the field can contain any command to be executed.

The third field is the terminal type normally connected to the line, as listed in termcap(5).

The remaining fields set flags in the `ty_status` entry, see `getttyent`, or specify a window system process to be maintained by `init` for the terminal line. The `on` and `off` options specify whether `init` is to execute the command in the second field. The `secure` option, used with `on`, allows root to login on the line. You can specify the `dialup` or `uucp` option after `secure`. The `dialup` option requires all users to enter a dialup password when logging on to that tty. The `uucp` option allows a user with UID uucp to log in to that tty. Ttys designated as `dialup` do not have to be named in the ttyd? format. Do not use quotation marks in the flag fields. The `window=` option is followed by a quoted command string that `init` executes before starting `getty`. If a line ends in a comment, the comment is included in the `ty_comment` field of the `ttyent` structure.

Example

```
console"/etc/getty std.1200"vt100   onsecure
tty00"/etc/getty d1200"vt100        ondialup# 555-1234
ttyh0"/etc/getty std.9600"hp2621-nlon# 254MC
ttyh1"/etc/getty std.9600"plugboardon# Sharon's office
ttyp0none        network            off
ttyp1none        network            off
ttyv0"/usr/new/xterm -L :0"vs100    onwindow="/usr/new/Xvs100 0"
tty01"/etc/getty std.9600"vt100     onsecuremark <engr> # eng tty
tty02"/etc/getty std.9600"vt100     onsecuremac !<engr> # mkt tty
```

The first line permits root login on the console at 1200 baud. The second line allows dial-up at 1200 baud without root login and requires a dialup password. The third and fourth lines allow logins at 9600 baud with terminal types of `hp2621-nl` and `plugboard`. The fifth and sixth lines are examples of network pseudo ttys, on which `getty` is not enabled. The seventh line is a terminal emulator and window system startup entry. The eighth line gives user mark and members of the `engr` group access to tty01. The ninth line gives user mac and all users except those in the `engr` group access to tty02.

Caveats

`getlogin` and the /etc/utmp file rely on the order of /etc/ttys. Changing the order (even inserting new entries in the middle of the file) invalidates some file entries (for example, /etc/utmp), causing `getlogin` to return erroneous results. Adding entries to the end of the file does not cause these problems.

If ConvexOS is in multiuser mode when you change /etc/ttys (for example, turn a teletype on or off or change a baud rate), you must use the ***`kill -1 1`*** signal to inform `init`.

See Also

/etc/utmp

gettytab(5), getty(8), login(1), on(1), off(1), getlogin(3), init(8)

/usr/adm/wtmp
login records

Format

Binary

Description

The /usr/adm/wtmp file records all logins and logouts. A null user name indicates a logout on the associated terminal (it is still easy to pair logins and logouts using the terminal name as the key). The terminal name ~ indicates that the system was rebooted at the indicated time.

The accounting program `ac` summarizes /usr/adm/wtmp and other accounting files.

Programs

`last, login, init, who, ac, telnetd, rlogind`

Caveats

The /usr/adm/wtmp file is not created by a program; if it is removed, login accounting ceases.

The file /usr/adm/wtmp has no size limits. To split the file, divide it at a `utmp` structure boundary. Another method is to rename /usr/adm/wtmp to /usr/adm/wtmp.old occasionally and copy /dev/null to /usr/adm/wtmp.

See Also

/usr/include/utmp.h

utmp(5)

Controller, device, and driver /ioconfig designations

C

Table 42 ConvexOS controller, device, and driver designations

/ioconfig designation	Description
ACM-001	Asynchronous Communication Local Terminal Controller, Multibus
ACM-002	
ACM-201	Asynchronous Communication Local Terminal Controller, VMEbus
COV-002	COVUE Driver, Multibus (*obsolete*)
COV-003	COVUE Driver, VMEBus (*obsolete*)
DKC-001	Disc Drive Controller, SMD, 2MB/Second Maximum, Multibus
DKC-002	Disc Drive Controller, SMD, 2.8MB/Second Maximum, Multibus
DKC-203	Disc Drive Controller, ESDI, 3MB/Second Maximum, VMEbus
DKC-204	Disc Drive Controller, SMD, 3MB/Second Maximum, VMEbus
DKC-IP2	Disc Drive Controller, IPI-2, PBus
DKD-001	Disc Drive, 10.5", 474MB, SMD, 1.85MB/Second, Multibus
DKD-002	Disc Drive, 14", 300MB, SMD, 1.2MB/Second, Multibus
DKD-005	Disc Drive, 9", 520MB, SMD, 1.86MB/Second, Multibus
DKD-006	Disc Drive, 9", 520MB, SMD, 1.86MB/Second, Multibus
DKD-008	Disc Drive, 9", 1.1GB, SMD, 2.46MB/Second, Multibus
DKD-206	Disc Drive, 9", 520MB, SMD, 1.86MB/Second, VMEbus
DKD-208	Disc Drive, 9", 1.1GB, SMD, 2.46MB/Second, VMEbus
DKD-214	Disc Drive, 5.25", 380MB, ESDI, 1.86MB/Second, VMEbus

Table 42 ConvexOS controller, device, and driver designations (continued)

/ioconfig designation	Description
DKD-281	Disc Drive, 8", 1.23GB, SMD, 3.02MB/Second, VMEbus
DKD-284	Disc Drive, 5.25", 780MB, ESDI, 2.46MB/Second, VMEbus
DKD-287	Disc Drive, 5.25", 1.53GB, ESDI, 2.92MB/Second, VMEbus
DKD-501	Disc Drive, 8", 1.23GB, IPI-2, 3.02MB/Second, Serial, PBus
DKD-502	Disc Drive, 8", 1.15GB, IPI-2, 6.00MB/Second, 2-Head Parallel, PBus
DKD-503	Disc Drive, 8", 3.22GB, IPI-2, 4.67MB/Second, Serial, PBus
DKD-504	Disc Drive, 8", 3.05GB, IPI-2, 9.35MB/Second, 2-Head Parallel, PBus
GPD-001	Graphics Display Device, VMEbus & Multibus
GPI-001	DR11-W Emulator, VMEbus & Multibus
HEC-001	High Speed Parallel (HSP) Controller, Echo Test (Echo-64), PBus
HEC-002	High Speed Parallel (HSP) Controller, User Written Device Driver (UWDD) 2, PBus
HEC-003	High Speed Parallel (HSP) Controller, User Written Device Driver (UWDD) 3, PBus
HEC-004	High Speed Parallel (HSP) Controller, User Written Device Driver (UWDD) 4, PBus
HEC-005	High Speed Parallel (HSP) Controller, User Written Device Driver (UWDD) 5, PBus
HED-001	High Speed Parallel (HSP) Device, User Written Device Driver (UWDD) 1, PBus
HED-002	High Speed Parallel (HSP) Device, User Written Device Driver (UWDD) 2, PBus
HED-003	High Speed Parallel (HSP) Device, User Written Device Driver (UWDD) 3, PBus
HED-004	High Speed Parallel (HSP) Device, User Written Device Driver (UWDD) 4, PBus
HED-005	High Speed Parallel (HSP) Device, User Written Device Driver (UWDD) 5, PBus
HiPPI	High Performance Peripheral Interface (HiPPI) Device Driver, PBus
HYP-001	HYPERChannel Device Driver, VMEbus & Multibus
LAN-001	Ethernet Communication Controller, Multibus

Table 42 ConvexOS controller, device, and driver designations (continued)

/ioconfig designation	Description
LAN-002	HYPERChannel Controller, Multibus
LAN-004	
LAN-006	X.25 Communication Controller, Multibus
LAN-007	Ethernet Communication Controller, VMEbus
LAN-202	UltraNetwork Communication Controller, VMEbus
LAN-204	HYPERChannel Controller, VMEbus
LAN-208	Fiber Distributed Data Interface (FDDI) Communication Controller, VMEbus
LAN-501	High Performance Peripheral Interface (HiPPI) Controller, PBus
MTC-001	Tape Drive Controller, STK Interface, Multibus
MTC-201	Tape Drive Controller, STK Interface, VMEbus
MTC-202	
MTC-B2X	Tape Library Interface (TLI) Controller, PBus
MTC-BMX	Tape Library Interface (TLI) Controller, Block Mux, PBus
MTC-IP3	Intelligent Tape Controller (ITC), IPI-3, PBus
MTD-001	Tape Drive, 50 IPS Start-Stop, STK Interface, Multibus
MTD-002	Tape Drive, 125 IPS Start-Stop, STK Interface, Multibus
MTD-003	Tape Drive, 200 IPS Start-Stop, STK Interface, Multibus
MTD-004	Tape Drive, 50 IPS Start-Stop/100 IPS Streaming, STK Interface, Multibus
MTD-201	Tape Drive, 50 IPS Start-Stop/100 IPS Streaming, STK Interface, VMEbus
MTD-202	Tape Drive, 125 IPS Start-Stop, STK Interface, VMEbus
MTD-203	Tape Drive, 200 IPS Start-Stop, STK Interface, VMEbus
MTD-204	Tape Drive, 50 IPS Start-Stop/100 IPS Streaming, STK Interface, VMEbus
MTD-207	Tape Drive, IBM 3480 Compatible, SCSI, VMEbus
MTD-208	Tape Drive, Digital Audio Tape (DAT), SCSI, Differential, VMEbus
MTD-217	Tape Drive, IBM 3480 Compatible, Automatic Cartridge Loader, SCSI, VMEbus

Table 42 ConvexOS controller, device, and driver designations (continued)

/ioconfig designation	Description
MTD-227	Tape Drive, IBM 3480 Compatible, IDRC Compatible Data Compression, SCSI, VMEbus
MTD-237	Tape Drive, IBM 3480 Compatible, IDRC Compatible Data Compression, Automatic Cartridge Loader, SCSI, VMEbus
MTD-238	Tape Drive, Digital Audio Tape (DAT), DCLZ (LZ2/LZW) Compatible Data Compression, Automatic Cartridge Loader, SCSI, Differential, VMEbus
MTD-301	Tape Drive, IBM 3480 Compatible, Block Mux, PBus
MTD-302	
MTD-321	Tape Drive, IBM 3480 Compatible, 4.5 MB/second, PBus
MTD-322	Tape Drive, IBM 3480 Compatible, 3 MB/second, PBus
MTD-341	Tape Drive, IBM 3480 Compatible, Block Mux, PBus
MTD-342	
MTD-R90	Tape Drive, IPI-3, PBus
NET-002	HYPERchannel Network Interface, Multibus
PLT-001	Plotter Device, Versatec Interface, Differential, Multibus
PLT-002	Plotter Device, Versatec Interface, TTL, Multibus
PRC-001	Line Printer Controller, Centronics Interface, Multibus
PRT-001	Line Printer, Centronics Interface, Multibus
PRT-CEN	Line Printer Controller & Device, Centronics Interface, VMEbus
PRT-DAT	Line Printer Controller & Device, Dataproducts Interface, VMEbus
SCI-001	X.25 Communication Controller, Multibus
SSC-000	Special Systems Controller Type 00
SSC-001	Special Systems Controller Type 01
SSC-002	Special Systems Controller Type 02
SSC-003	Special Systems Controller Type 03
SSC-004	Special Systems Controller Type 04
SSC-005	Special Systems Controller Type 05

Table 42 ConvexOS controller, device, and driver designations (continued)

/ioconfig designation	Description
SSC-006	Special Systems Controller Type 06
SSC-007	Special Systems Controller Type 07
SSC-008	Special Systems Controller Type 08
SSC-009	Special Systems Controller Type 09
SSC-010	Special Systems Controller Type 10
SSC-011	Special Systems Controller Type 11
SSC-012	Special Systems Controller Type 12
SSC-013	Special Systems Controller Type 13
SSC-014	Special Systems Controller Type 14
SSC-015	Special Systems Controller Type 15
SSU-000	Special Systems Unit (Device) Type 00
SSU-001	Special Systems Unit (Device) Type 01
SSU-002	Special Systems Unit (Device) Type 02
SSU-003	Special Systems Unit (Device) Type 03
SSU-004	Special Systems Unit (Device) Type 04
SSU-005	Special Systems Unit (Device) Type 05
SSU-006	Special Systems Unit (Device) Type 06
SSU-007	Special Systems Unit (Device) Type 07
SSU-008	Special Systems Unit (Device) Type 08
SSU-009	Special Systems Unit (Device) Type 09
SSU-010	Special Systems Unit (Device) Type 10
SSU-011	Special Systems Unit (Device) Type 11
SSU-012	Special Systems Unit (Device) Type 12
SSU-013	Special Systems Unit (Device) Type 13
SSU-014	Special Systems Unit (Device) Type 14
SSU-015	Special Systems Unit (Device) Type 15

Table 42 ConvexOS controller, device, and driver designations (continued)

/ioconfig designation	Description
TTY	Communication Terminal Device, VMEbus & Multibus
UDD-001	User Debug Driver (UDD) 1 for User Written Device Driver (UWDD) debug
UDD-002	User Debug Driver (UDD) 2 for User Written Device Driver (UWDD) debug
USC-001	User-specified Controller Type 1
USC-002	User-specified Controller Type 2
USC-003	User-specified Controller Type 3
USC-004	User-specified Controller Type 4
USC-005	User-specified Controller Type 5
USC-006	User-specified Controller Type 6
USC-007	User-specified Controller Type 7
USC-008	User-specified Controller Type 8
USC-009	User-specified Controller Type 9
USD-001	User-specified Device Type 1
USD-002	User-specified Device Type 2
USD-003	User-specified Device Type 3
USD-004	User-specified Device Type 4
USD-005	User-specified Device Type 5
USD-006	User-specified Device Type 6
USD-007	User-specified Device Type 7
USD-008	User-specified Device Type 8
USD-009	User-specified Device Type 9
VER-001	Plotter Controller, Versatec Interface, Differential, Multibus
VER-002	Plotter Controller, Versatec Interface, TTL, Multibus
VPC-001	Plotter Controller, Versatec Interface, VMEbus
VPD-001	Plotter Device, Versatec Interface, VMEbus

Reporting problems

D

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 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 162 illustrates the use of `which` to find the full path name of a ficticious program called `filefix`.

Figure 162 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 163 illustrates the use of `vers` to find the version number of the program `filefix`.

Figure 163 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 164 is an example of a .contact file viewed using the cat command

Figure 164 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 11 on page 419.

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 12 on page 420. 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 12 on page 420), `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 1 on page 415.

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 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 412.

contact can be invoked as shown in Figure 165.

Figure 165 Beginning a contact report

```
% contact
Welcome to contact version 0.12 (90/02/05)
```

After invoking contact, the system responds with a welcome message and a series of questions regarding your hardware, software, or documentation question. 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 2 Enter information about yourself.

After invoking `contact` you are asked for some information about yourself, as shown in Figure 166, 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 412)

Figure 167 illustrates how to invoke `contact` and how the system responds if you have a `.contact` file in your home directory. If you have a .contact file, go to Step 3.

Figure 166 Beginning a `contact` report without a .contact file

```
% contact
Welcome to contact version 0.12 (90/02/05)

Enter your name, title, phone number, and corporate name (^D to terminate)
> Chris Smith
> Programmer
> (214) 900-2000
> Jupiter Corporation
> ^D
```

Step 3 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 167. Enter the name of the product.

Figure 167 Beginning a `contact` report with a `.contact` file

```
% contact
Welcome to contact version 0.12 (90/02/05)

Enter the name of the product involved
> filefix
```

Step 4 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 168.

Figure 168 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 411. 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 5 Enter a one-line problem summary.

The `contact` utility prompts for a one-line summary of the problem, as shown in Figure 169.

Figure 169 Prompt for a short problem summary

```
Enter a short (1 line) summary of the problem
> Trouble suspending filefix
```

This summary is the subject header in any further correspondence regarding your problem. Please make this summary as descriptive as possible in one line.

Step 6 Enter a detailed problem description.

The `contact` utility prompts for a detailed description of the problem, as shown in Figure 170.

Figure 170 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
```

Enter a detailed description of the problem. When you have completed your description, press **CTRL-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.

Step 7 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 171 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 171 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.
> 4
```

Step 8 Give instructions how to reproduce the problem.

`contact` prompts for instructions on how to reproduce the problem, as shown in Figure 172.

Figure 172 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 9 Give applicable comments.

The `contact` utility prompts for any other pertinent comments, as shown in Figure 173. Please include all relevant information.

Figure 173 Prompt for additional, applicable comments

```
Enter any comments that are applicable(^D to terminate)
> Perhaps I am using the wrong command?
> ^D
```

Step 10 Offer suggestions for documentation and support.

The `contact` utility prompts for suggestions regarding documentation supporting the product, as shown in Figure 174.

Figure 174 Prompt for suggestions or comments

```
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 11 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 175.

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 175 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 12 To finish your `contact` report, you are given the choice to review, edit, submit, or abort the report, as shown in Figure 176.

You must enter the number associated with your selection.

Figure 176 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 176 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.

Master index

This index covers four books. Prefixes with the page numbers in each index entry indicate the book in which the related information can be found. Prefixes are associated with individual books in the following way:

CO- *Managing ConvexOS: Configuration Guide*

OP- *Managing ConvexOS: Operations Guide*

TM- *ConvexOS Tape System Manager's Guide*

TO- *ConvexOS Tape System Operator's Guide*

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

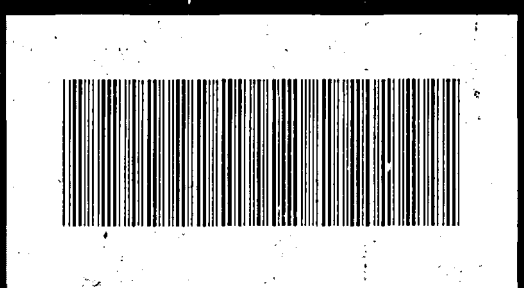
T

U

V

W

X



Order Number
DSW-030

Document Number
710-001430-211