File System Maintenance [4]

This chapter includes the following topics for all Cray Research systems:

- Mounting and unmounting file systems
- File system utilities
- File system backup and restoration
- File system checking and repair with `fsck`(8)

4.1 Mounting and unmounting file systems

A disk device is a sequential array of data until it is mounted. When the device is mounted, the UNICOS kernel interprets the data as a UNICOS file system and makes the file system available as part of the system's complete directory structure.

File systems are mounted with the `mount`(8) command. The start-up script `/etc/rc` (see `brc`(8)) traditionally mounts the various file systems that are available during system operation. It may prove more convenient to have `/etc/rc` execute another script or set of scripts (located in the `/etc` directory) to mount the various file systems, allowing system users to use the same scripts to mount the file systems during single-user mode.

The `/etc/fstab` file (see `fstab`(5)) contains descriptions of file systems and swapping partitions. The `mount` command searches this file, if present, to determine the parameters it should use.

File systems are unmounted from the system by using the `/etc/umount` command (see `mount`(8)). When unmounting clustered file systems, you must specify the directory on which the file system is mounted; `umount` does not work if the file system is a cluster and you specify the cluster descriptor file name. You may use the node name to unmount a regular file system.

The `umount`(8) command flushes the file system cache to the disk before actually unmounting the file system.

4.2 File system utilities

The following utilities are associated with the maintenance of file systems and files (more information about these commands can be found in UNICOS man pages).

Command	Description
`bmap`(8)	Reports the current use of a given block in a file system. Used to determine whether a block that needs to be flawed is currently in use by a file and, if so, the name of the file.
`ddstat`(8)	Displays configuration information about disk type character and block special devices.
`df`(1)	Reports the number of free blocks available for mounted file systems.
`diskusg`(8)	Summarizes the disk usage on a file system by file ownership.
`dmap`(8)	Reports the slices of all physical devices that compose a given logical device or the slices of all logical devices that reside on a given physical device.
`du`(1)	Provides a summary of the disk use on a file system by directory structure.
`fck`(1)	Displays data block allocation for a specific file.
`fsck`(8)	Checks and corrects the consistency of a file system before it is mounted. File system checking should always be a part of the system startup procedures. Use of the `fsck` command is described in detail on Section 4.4.1, page 99.
`fsed`(8)	Debugs file systems.
`fsmap`(8)	Reports all free block areas in a specific file system; useful for determining fragmentation.
`fstest`(8)	Tests file systems and disk devices.
`labelit`(8)	Reads or writes file system labels and security levels.

`ldcache`(8)	Assigns and displays logical device cache.
`mkfs`(8)	Creates a file system on a logical device.
`mknod`(8)	Builds a directory entry and inode for a special file.
`mkspice`(8)	Creates physical disk device inodes that describe the spare sector map, factory flaw map, and diagnostic and customer engineering slices.
`sdconf`(8)	Controls the state of a disk drive.
`sdstat`(8)	Displays information about disk device I/O.
`setf`(1)	Initializes a new or existing file. The file is created if it does not already exist, and a specified number of bytes or blocks are allocated to it.
`setfs`(8)	Modify attributes of a file system after creation.

4.3 File system backup and restoration

This section describes some of the procedures you can use to back up files and file systems on Cray Research systems running UNICOS.

The following sections discuss the three major applications of the file system backup and restoration tools:

- Local backup of file systems (usually to tape), using the `dump`(8) and `restore`(8) commands
- Remote backup of file systems through the network to another server by using the `rdump`(8) and `rrestore`(8) commands

4.3.1 Local backup

The UNICOS `dump`(8) and `restore`(8) utilities provide the capability to backup and reload file systems. This is usually done using tapes. This section assumes that your Cray Research computer system has a tape subsystem as the target for the file system dump.

Warning: Use of the `dump`(8) and `restore`(8) utilities on a Trusted UNICOS system requires a multilevel archive medium.

4.3.1.1 Using the dump command

The `dump` command copies to magnetic tape all files changed after a specified date in a specified file system. Refer to the `dump`(8) man page for a complete list of the options available for use with `dump`. Some of the most useful options are described in this section.

The `-t` *dump_type* option and argument specify the dump level; *dump_type* can be a number from 0 through 9. If the `-t` option is omitted, `dump` defaults to a level-9 dump. A level-0 dump is a complete dump; all files in a file system are copied to tape. For a given dump level *x*, only those files modified since the last dump of level- *y* (*y* <= *x*) are dumped.

For example, if you did a level-2 dump on Monday, followed by a level-4 dump on Tuesday, a subsequent level-3 dump on Wednesday would contain all files modified or added since Monday.

Although this arrangement provides significant flexibility in scheduling dumps, a relatively simple scheme is recommended:

- Once a week, perform a level-0 dump of each file system. Use at least two sets of tapes so that you can recover files even if a file system is destroyed while you are dumping it. Because `dump` opens and reads a raw file system (instead of using the operating system to open and read each file), it is recommended that at least this complete dump be performed on a quiet system, with no users other than the administrator or operator logged in. `dump` can read an unmounted file system; if you prefer, you can unmount the file system to be dumped before you begin.
- On each day that you do not perform the level-0 dump, perform a level-9 dump of each file system. Use a different set of tapes each day for two weeks to safeguard your data. Each level-9 dump contains the files modified since the last weekly level-0 dump. Thus, to reload a file system, you need only two sets of tapes: the weekly dump, and the latest daily dump.

To simplify the task of performing dumps, you can set up a shell script for your operator as follows:

```
if [ "$1" = "daily" ] ; then
  level=9
elif [ "$1" = "weekly" ] ; then
  level=0
else
  echo "Usage: backup daily|weekly"
  exit 1
fi
```

```
for fs in /dev/dsk/root /dev/dsk/usr /dev/dsk/slash_u ; do
  /etc/dump -t level -u $fs
done
```

Using this script, the operator needs to enter only `backup daily` for a daily backup, or `backup weekly` for a weekly backup.

The `-u` option causes `dump` to write the date and time of the beginning of the dump in the `/etc/dumpdates` file, which contains the file system name, the level of dump, and the time and date that the dump started. (Refer to `dump`(5) for the format of the `dumpdates` file.)

The following options can be used with `dump`(8) to specify different tape attributes:

`-l` Specifies labeling of the tapes. The following values can be used with the `-l` option:

`nl`	Nonlabeled tapes
`sl`	IBM standard labeled tapes
`al`	ANSI labeled tapes

`-v` Allows you to enter a list of volume serial numbers (VSNs). If this option is omitted, you are asked to type in a VSN list; the VSNs in the list are separated by colons (:). Each VSN is a string consisting of 1 to 6 characters. In the following example, `dump` would use the volumes `r1`, `r2`, and `r3`:

```
/etc/dump -t 0 -u -v r1:r2:r3 /dev/dsk/root
```

`-D` Allows you to request a specific tape device for the dump, as in the following example:

```
/etc/dump -t 2 -u -D tape00 /dev/dsk/usr
```

Refer to the `tpmnt`(1) command for a more complete description of these tape-specific options. The `dump` command actually performs an `rsv`(1) and `tpmnt`(1) from these specifications.

You may want to perform your own `rsv` and `tpmnt` commands before the dump, as in the following example:

```
rsv TAPE 1
tpmnt -l sl -P /tmp/name -v r1:r2:r3
/etc/dump -t0 -u -f /tmp/name /dev/dsk/root
rls -a
```

You may also use a disk file, rather than tape, for some special purpose, by specifying the `-f` option, as in the following example:

```
/etc/dump -t 9 -f /tmp/dumpfile /dev/dsk/usr
```

Refer to the `dump`(8) man page for a complete list of the options available for use with the `dump` command.

Warning: Use of the `dump`(8) and `restore`(8) utilities on a Trusted UNICOS system requires a multilevel archive medium.

4.3.1.2 Using the `restore` command

You can reload a file system from the dump tapes by using the `restore`(8) command. The `restore` command accepts various options. Refer to the `restore`(8) man page for a complete list of the options available for use with `restore`. Some of the most useful options are described in this section.

To reload a file system from dump tapes, first use the `mkfs`(8) command to initialize the unmounted file system. Mount the file system and change directories (using `cd`(1)) to the mount point. Next, use `restore` with the `-r` option to reload the full (level-0) dump; use `restore` with the `-r` option again to reload the incremental (level-9) dump.

The `restore` command requires many synchronous write operations, which can be time-consuming. You can disable synchronous write operations and increase the efficiency of the `restore` command by using the `ldcache`(8) facility.

The following examples show how you would perform a full restore operation if you had a file system that had been destroyed. First, you would initialize the unmounted file system, `/newfs`; then you would mount the file system, change directories to the mount point, and reload the full dump, as follows:

```
/etc/mkfs /dev/dsk/newfs
/etc/mount /dev/dsk/newfs /newfs
cd /newfs
/etc/restore -r -V fs1:fs2:fs3
```

The `-V` option specifies a volume serial number (VSN) list of dump tapes to be used for the restore operation. In this example, the tapes with VSNs of `fs1`, `fs2`, and `fs3` are used.

After this completes, you would restore the last incremental dump, as follows:

```
/etc/restore -r -V fsd1
```

Now, you would remove the `restoresymtab` file (this file is created to pass along information between the restore of the complete dump and incremental dump) by using the following command:

```
rm restoresymtab
```

The new file system would have all of the files in it up to the last incremental dump.

If you want to reload a particular set of files from a dump tape, use the following invocation of `restore`:

`/etc/restore -x` *filenames*

The `-x` option causes `restore` to extract named files from the tapes. The file names are relative to the mount point of the file system.

For example, if you had a file system `/dev/dsk/usr_mail` mounted on `/usr/mail`, you would dump it by using the following command:

```
/etc/dump -t 0 -u /dev/dsk/usr_mail
```

To reload `/usr/mail/fred` and `/usr/mail/root` from this dump, you would first change directories to `/usr/mail`, and then use `restore` with the `-x` option to reload those particular files, as follows:

```
cd /usr/mail
/etc/restore -x fred root
```

The `restore` command can also be used in interactive mode. After reading in the directory information, `restore` provides a shell-like interface that allows you to move around the directory tree, selecting files for extraction. You can use the commands `ls`(1), `cd`(1), and `pwd`(1) as they are used in the shell and add or delete files as you wish. This is a convenient way to examine the contents of a dump tape and restore one or more single files or directories.

The `restore` command also has a `-t` option that writes out the table of contents of the dump tape to standard output.

As with `dump`, you can perform your own `rsv`(1) and `tpmnt`(1) first, using the `-f` option, as in the following example:

```
rsv CART
tpmnt -l sl -v r1:r2:r3 -g CART -P /tmp/tape
/etc/restore -if /tmp/tape
```

Select files to extract and quit the interactive mode

```
rls -a
```

The `-F` option of the `restore` command specifies the tape file ID of the dump tape to be restored. The default ID is the volume serial number (VSN) of the first tape of the dump file.

You must follow some special procedures if your `/usr` or `/` (root) file system is destroyed, because you need certain files and directories to use the `restore` command and the online tape daemon. The default installed tape daemon uses the `/usr/spool/tape` directory, and the message daemon uses the `/usr/spool/msg` and `/usr/adm/msg` directories. If the tape daemon or message daemon have not been installed by default, other directories may be used. Directories used by the daemons must exist so that the tape and message daemons can add files to them before you restore the `/usr` file system.

When restoring `/` (the root file system), you also need the tape daemon and message daemon. If a lack of disk space prevents you from keeping a spare root file system, you must keep the following binary files on the operator's workstation on Cray PVP systems.

```
/etc/msgdstop
/etc/tpapm
/etc/tpbmx
/etc/tpclr
/etc/tpdev
/etc/tpconfig
/etc/tpdstop
/etc/tpfrls
/etc/tpgstat
/etc/tplabel
/etc/tpmls
/etc/tpmql
/etc/tpset
/etc/tpu
/usr/lib/tp/avrproc
/usr/lib/tp/clsfile
/usr/lib/tp/fesreq
/usr/lib/tp/flush
/usr/lib/tp/openfile
/usr/lib/tp/opentdt
/usr/lib/tp/proceot
/usr/lib/tp/proceov
/usr/lib/tp/readerr
/usr/lib/tp/readvol
```

```
/usr/lib/tp/slnet
/usr/lib/tp/stknet
/usr/lib/tp/tpdaemon
/usr/lib/tp/tppos
/usr/lib/tp/writeerr
/usr/lib/tp/writevol
/usr/lib/msg/infd
/usr/lib/msg/msgd
/usr/lib/msg/msgdaemon
/usr/lib/msg/oper
/usr/lib/msg/rep

/bin/rls
/bin/rsv
/bin/tpcatalog
/bin/tplist
/bin/tpmnt
/bin/tprst
/bin/tpscr
/bin/tpstat
```

The following character special file for tapes is also needed for restoring directories and files, and it should be kept on the operator's workstation on Cray PVP systems.

```
/dev/bmxdem
```

Any other files you need for deadstarting the system should also be kept on the boot media.

You will need the following files to run `restore` when the system is running:

```
/usr/lib/msg/msgdaemon
/usr/lib/tp/tpdaemon
```

Copy these files into your new root file system, and then start the tape and message daemons. Now you should be ready to proceed with the restoration of your root dump tapes.

If you are running an autoloader, you must have a full system, which includes all the basic files in the major directories of your system, in order to run the `restore` command successfully. Therefore, the procedure outlined in this section will not provide you with everything you need.

4.3.2 Remote backup

This section describes how to perform file system dumps and restorations on machines that do not have Cray Research online tapes but are part of a TCP/IP network.

Warning: This feature is not supported on a Cray ML-Safe configuration of the UNICOS operating system.

The `rdump`(8) and `rrestore`(8) commands are used to perform file system backups to a tape device on a remote host. These commands provide an I/O interface between the `dump` and `restore` commands on the Cray Research mainframe and the device on the remote host. The commands create a remote server process running the `/etc/rmt` command on the client machine. This processes accesses the tape device.

You must log in as `root` to perform the backup and restore procedures. Because the file transfer is performed across the network, you must ensure that the user `root` on the remote host has the Cray Research mainframe listed as a target machine in the `.rhosts` file.

For example, if you (as root) want to dump a file system from the Cray Research mainframe named `sn1203` to the remote server `hall`, be sure that a `.rhosts` file exists in the root directory on `hall` and that it contains an entry for `sn1203`. For more information about `.rhosts` files, see the `rhosts`(5) man page.

The following examples illustrate the use of `rdump` and `rrestore`.

Example 1: The following command performs a level 0 dump of file system `fs1` to the tape device `rst0` on the host `host1`:

```
rdump -f host1:/dev/rst0 - -t 0 /dev/dsk/fs1
```

Example 2: The dump performed in example 1 can be restored to a file system `filesys2` on the Cray Research system with the following command:

```
rrestore -f host1:/dev/rst0 - -x
```

Refer to the `rdump`(8), `rrestore`(8), `dump`(8), and `restore`(8) man pages for complete descriptions of these commands and their options. The `fsck`(8) command is an indispensable tool that maintains the consistency of UNICOS file systems by interactively repairing most file system damage resulting from an operating system crash. `fsck` reports its progress through a series of phases, checking a file system for consistency and repairing any inconsistencies it discovers. If `fsck` determines that a file system has no inconsistencies, or that it has had its inconsistencies repaired, you can safely mount the file system.

You should also use `fsck` to ensure that file systems are not damaged before going into multiuser mode or doing backups.

4.4 File system checking and repair with `fsck`

The following section provides an overview of file system operation, and how using the `fsck`(8) command can help ensure date integrity. The subsequent sections describe the behavior of `fsck` and the phases that `fsck` goes through while checking a file system.

4.4.1 Overview of file system operation

The file system directories consist of pointers to inodes. In turn, these inodes point to blocks of pointers to data and directories. Unfortunately, the operating system cannot perform extensive validation of file system integrity, and when this elaborate construction loses its consistency, there can be a serious loss of data. With careful maintenance, however, you can ensure that the file system works safely and efficiently.

Damage to the file system occurs when a portion of its structure is lost before it can be written to disk. This damage is typically caused by a hardware, operational, or operating system failure. While a file system is in use, it consists of a combination of data on disk and, for efficiency, data in kernel memory.

The memory-resident data is written at regular intervals by the `init`(8) process with the `sync`(2) system call. The cache can be written to disk at any time by using the `sync`(1) command. On Cray PVP systems that use logical device caching, the logical device cache can be written to disk at any time by using the `ldsync`(8) command. (`ldsync` should always be issued after the `sync` command is issued.)

The `inittab`(5) file allows you to control the rate of the `sync` and `ldsync` operations with the `sleeptime` and `ldsynctm` entries. More frequent execution of the `ldsync` command reduces the risk of file system corruption in the event of a system crash, but the increased system overhead may impact system performance. The default rate of `ldsync` is 120 seconds, and the default rate of `sync` is 30 seconds.

If the operating system is stopped before all the data is returned to disk, the structure of the file system may be damaged. Usually, this damage is corrected by `fsck` at restart time. If the file system is used in an inconsistent state, however, the damage quickly spreads throughout the system and destroys it.

The rules for correct use of the file system are as follows:

- Before using a file system, you must ensure that it is consistent
- When you stop the system, each file system must be consistent

The `fsck`(8) command checks and corrects the consistency of a file system before it is mounted. File system checking should always be part of the system startup procedures.

4.4.2 Using `fsck`

When using the `fsck`(8) command, usually you will respond `yes` to all of the prompts. However, in the event of a system crash, the damage may be extensive enough to warrant recovery from a back-up tape. If the file system is consistent, `fsck` prints a summary of statistics about the file system.

Note: Many corrective actions may result in some data loss.

You can use the `-y` option on the `fsck` command to avoid the questions asked by `fsck` concerning inconsistencies it found. This option automatically attempts repairs as though you had answered `yes` to the questions. Use this with caution, however, because the corrections may involve some data loss.

The following examples show the results of the use of `fsck`, first, without the `-y` option, and then with the `-y` option.

```
$ /etc/fsck tmp2fs
tmp2fs: device tmp2fs opened as partition 0
tmp2fs: superblock fname , fpack
tmp2fs: Phase 1 - Check Blocks and Sizes
tmp2fs: Phase 2 - Visit Directories
tmp2fs: Phase 3 - Checking Directories
tmp2fs: Phase 4 - Checking Non-Directories and Link Counts
tmp2fs:
tmp2fs: i-node 0.0000002 has problems
tmp2fs: i-node summary
tmp2fs:         owner 0, mode 100644, link 1
tmp2fs:         size 13312, mtime Wed Dec 31 18:00:00 1989
tmp2fs: paths to this i-node ...
tmp2fs:          - ./afile
tmp2fs:         out-of-range allocations
tmp2fs:         (warning) file size field in i-node is incorrect
tmp2fs: clear ('y' or 'n')? y
tmp2fs: Phase 5 - Verify Dynamic Information
tmp2fs: block 0/71 missing from free block list
tmp2fs: block 0/72 missing from free block list
tmp2fs: block 0/73 missing from free block list
tmp2fs: partition 0, free i-node count is 62, should be 63
tmp2fs: rebuild dynamic information ('y' or 'n')? y
tmp2fs: Phase 6 - Rebuilding Dynamic Information
tmp2fs: file system summary
tmp2fs:     partition 0 on device tmp2fs
tmp2fs:         64 total i-nodes (63 free i-nodes)
tmp2fs:         30 total tracks (25 free tracks, 8 free blocks)
tmp2fs: ***** FILE SYSTEM WAS MODIFIED *****
```

```
$ /etc/fsck -y tmp2fs
tmp2fs: device tmp2fs opened as partition 0
tmp2fs: superblock fname , fpack
tmp2fs: Phase 1 - Check Blocks and Sizes
tmp2fs: Phase 2 - Visit Directories
tmp2fs: Phase 3 - Checking Directories
tmp2fs: Phase 4 - Checking Non-Directories and Link Counts
tmp2fs:
tmp2fs: i-node 0.0000002 has problems
tmp2fs: i-node summary
tmp2fs:         owner 0, mode 100644, link 1
tmp2fs:         size 13312, mtime Wed Dec 31 18:00:00 1989
tmp2fs: paths to this i-node ...
tmp2fs:          - ./afile
tmp2fs:         out-of-range allocations
tmp2fs:         (warning) file size field in i-node is incorrect
tmp2fs: clear? yes
tmp2fs: Phase 5 - Verify Dynamic Information
tmp2fs: block 0/71 missing from free block list
tmp2fs: block 0/72 missing from free block list
tmp2fs: block 0/73 missing from free block list
tmp2fs: partition 0, free i-node count is 62, should be 63
tmp2fs: rebuild dynamic information? yes
tmp2fs: Phase 6 - Rebuilding Dynamic Information
tmp2fs: file system summary
tmp2fs:     partition 0 on device tmp2fs
tmp2fs:         64 total i-nodes (63 free i-nodes)
tmp2fs:         30 total tracks (25 free tracks, 8 free blocks)
tmp2fs: ***** FILE SYSTEM WAS MODIFIED *****
```

All mountable file systems should be listed in the `fstab`(5) file, which the `mfsck`(8) command uses, and you should check these file systems each time the system is rebooted.

The `fsck` command cannot be executed on a mounted file system, because this would repair only the physical disk, leaving all the system buffers incorrect. The only exception to this is the `root` file system, which is always mounted and must be repaired while mounted.

To repair the `root` file system, enter the following command:

```
fsck /dev/dsk/root
```

`fsck` may respond by asking questions. You can use `mfsck` to run file system checks in parallel, which can speed system startup. If `fsck` prompts for a response to any problem, an analyst experienced in repairing UNICOS file systems should assist you. You may reply `n` to any `fsck` prompt, leaving the indicated inconsistency unresolved.

Under no circumstances should you mount or boot from any file system that still has unresolved inconsistencies detected by `fsck`.

When you are using `fsck` to repair damage following a crash, it is useful to first use the `-n` option with `fsck` to survey the damage (the `-n` option assumes an automatic `no` response to all questions). Having seen the extent of the damage and determined that there are no extraordinary inconsistencies, you may use the `-y` option, in conjunction with other `fsck` options, to avoid having to type `y` in response to each question.

The `fsck` command provides orphan checking, which allows the possible recovery of files that have no links to the file system directory structure. (Such a file may occur when a directory entry for the file is not written to the disk prior to a system crash.)

Note that the set-user-ID and set-group-ID bits are cleared on all orphans recovered by `fsck`.

4.4.3 `fsck` phases

The `fsck`(8) command goes through nine phases, described in the following sections. Unless otherwise specified, the phases are the same for all native Cray Research file systems.

4.4.3.1 Initialization phase

During the initialization phase, `fsck` verifies that all opened devices are partitions of the same file system. Most problems cause the program to stop. The following checks are performed:

- All superblocks and dynamic blocks can be located
- Inode blocks, bad blocks, super blocks, and dynamic blocks can be allocated without error
- Track size and root inode number are consistent across all partitions in the file system
- Total inode count is consistent with the number of blocks allocated for inodes

- Track size and total size are consistent with the value returned by a `stat`(2) system call

4.4.3.2 Phase 1

During phase 1, `fsck` examines each active inode. Errors detected during this phase will be announced during phase 3 or 4 when a file name is associated with the inode.

The following checks are performed:

- The mode field is valid
- The sectors belonging to the file may be allocated without conflict and within valid areas of the file system
- The last byte of the file is contained in an allocated sector

4.4.3.3 Phase 2

If the `-f` option is not specified, `fsck` enters phase 2 and examines the contents of all directory sectors, noting garbled sectors and inodes that are not valid. Links from the directory to inodes are saved. The `fsck` command travels through the directories by order of the address of the first disk block, rather than in directory tree search order. By doing so, the execution speed is faster, but a phase 3 subroutine is required to gather a file name for a problem report.

For each entry in a directory, `fsck` verifies that a nonzero inode field refers to an accessible inode.

The directory tree structure is validated. Each directory should be accessible along only one path, and it should contain valid `.` (dot) and `..` (dot dot) entries.

4.4.3.4 Phase 2X

If the `-f` option is specified, `fsck` skips phases 2 through 4 and prints the errors discovered during phase 1, but offers no opportunity to clear the inode.

4.4.3.5 Phase 3

The `fsck` command reports errors in directory inodes. The names of unlinked directories are displayed to the operator to be selected for relinking and

directories with garbled sectors are displayed to the operator to be selected for clearing.

4.4.3.6 Phase 4

All nondirectory inodes with problems are reported in phase 4.

4.4.3.7 Phase 5

During phase 5, `fsck` examines the free track, free sector, and free inode information in the dynamic block. Any errors detected are announced. The operator is offered an opportunity to rebuild the dynamic block.

4.4.3.8 Phase 6

During phase 6, `fsck` rebuilds all information in the dynamic block, except the magic word.

4.4.3.9 Termination phase

A summary of the state of the file system is printed.