Managing File Systems in Oracle® Solaris 11.3

Part No: E54785
October 2017

Managing File Systems in Oracle Solaris 11.3

Part No: E54785

Contents

Using This Documentation

- **Overview** – Provides an overview of Oracle Solaris file systems, which includes information about how to manage one or more file systems and perform file system administration tasks.
- **Audience** – System administrators.
- **Required knowledge** – Basic Oracle Solaris or UNIX system administration experience and general file system administration experience.

Product Documentation Library

Documentation and resources for this product and related products are available at `http://www.oracle.com/pls/topic/lookup?ctx=E53394-01`.

Feedback

Provide feedback about this documentation at `http://www.oracle.com/goto/docfeedback`.

♦ ♦ ♦ C H A P T E R 1

About Managing File Systems

Managing file systems, data, free space, file system backup, and restore are some of the most important system administration tasks. This chapters provides an overview of how files and directories are organized to form a file system, and describes different types of file system supported by Oracle Solaris.

This chapter contains the following topics:

- "About File Systems" on page 9
- "Supported File Systems" on page 10
- "Extended File Attributes" on page 13

To familiarize with the common terminologies related to Oracle Solaris data and storage, see Glossary in *Managing SAN Devices and Multipathing in Oracle Solaris 11.3*.

About File Systems

It is common to have all the files organized in a single structure of directories, called as a file system tree in a UNIX environment. The directory structure starts with a root directory (`/`).

Files can be stored on different types of media or networks. The UNIX system enables you to attach and detach files stored on a given media. The process of attaching and detaching files from the directory is called mounting and unmounting of files. The directory, where a subtree or a file from one media is attached to the complete directory structure, is called a mount point.

A file system is a structure of directories that is used to organize and store files. Any of the items from the following list can be termed as a file system:

- Particular type of file system: disk-based, network-based, or virtual
- Entire file system tree, beginning with the root (`/`) directory
- Data structure of a disk slice or other media storage device

- Portion of a file tree structure that is attached to a mount point on the main file tree so that the files are accessible

The Oracle Solaris OS uses the virtual file system (VFS) architecture, which provides a common interface for various file system types. The VFS architecture enables the kernel to provide a common interface for usual operations such as reading, writing, and listing files. The execution of these operations are managed by kernel modules for a specific file system. The VFS architecture also makes it easier to add new file systems.

Supported File Systems

Oracle Solaris operating system (OS) supports file systems stored on a locally attached storage media such as, a hard disk and file systems stored on a network. Storing file systems locally or on a network enables the VFS architecture to provide information from kernel.

Disk-Based File Systems

Disk-based file systems enable you to access files stored on a locally attached media such as hard disks, CD-ROMs, or USB devices. Oracle Solaris supports different types of disk-based file systems.

UFS	Legacy UNIX file system (based on the BSD Fat Fast File system).
ZFS	Zettabyte file system (ZFS) is the default disk-based and root file system. The ZFS file system has some features which are not found in any other file system, such as, creating storage pools, snapshots, and using copy-on write semantics. These features are not available in any other file system. For more information about ZFS file system, see *Managing ZFS File Systems in Oracle Solaris 11.3*.
HSFS	High Sierra File System (HSFS) is used on CD-ROMs and DVDs. Oracle Solaris supports file systems created according to ECMA-119, ISO 9660:1998 or ISO 9660:1999 with Rock Ridge and Joliet extensions. The extensions provide additional features and attributes to files such as long file names (up to 255 characters), and UNIX permissions and attributes. Due to the nature of the media, a HSFS file system is read only.
PCFS	PC file system (PCFS), which enables read and write access to data and programs on DOS-formatted disks that are written for DOS-based personal computers.

UDFS	The Universal Disk Format (UDFS) file system which is the industry-standard format for storing information on the optical media technology called DVD (Digital Versatile Disc or Digital Video Disc).
SAM-QFS	SAM-QFS is an integrated hierarchical storage manager (HSM) and storage area network (SAN) file system. SAM is a component of HSM storage and archive management. QFS is the SAN scalable high performance file system component. SAM-QFS also has an integrated disk volume management and tape volume management. QFS also has a write once, read many times (WORM) file system capability. QFS can be used independently of SAM when just a file system is needed. SAM requires QFS and cannot be used independently of QFS. For more information, see Oracle Hierarchical Storage Manager and StorageTek QFS Software Installation and Configuration Guide Release 6.0.

Network-Based File Systems

Network-based file systems can be accessed from a network. Typically, network-based file systems reside on a server, and are accessed by other systems across the network. The different types of network-based file systems are as follows:

NFS	Network File System (NFS) is a common resource sharing service in an UNIX environment. With the NFS service, you can provide distributed resources (files or directories) by sharing them from a server and mounting them on multiple clients. Oracle Solaris support versions 2, 3 and 4 of the NFS protocol. For more information, see *Managing Network File Systems in Oracle Solaris 11.3*.
SMB or CIFS	Server Message Block (SMB) protocol is a resource sharing service used primarily in workgroups or domains of Microsoft Windows systems. With the Oracle SMB service, you can provide distributed resources (files or directories) to Windows and Mac OS systems by sharing them from a server and mounting them on multiple clients. For more information, see *Managing SMB File Sharing and Windows Interoperability in Oracle Solaris 11.3*.

Virtual File Systems

Virtual file systems are memory-based file systems that provide access to special kernel information and facilities. Most virtual file systems do not use file system disk space. Also, some virtual file systems, such as the temporary file system (TMPFS) use the swap space on a disk. The different types of virtual file system are as follows:

CTFS	The contract file system (CTFS) is an interface for creating, controlling, and observing contracts. A contract enhances the relationship between a process and a system resource. It provides richer error reporting and means of delaying the removal of a resource. The service management facility (SMF) uses process contracts (a type of contract) to track the processes, which compose a service. Using process contracts helps SMF to identify service failure even if a part of a multi-process service fails.
MNTFS	The `mnttab` file system (MNTFS) provides read-only access to the table of mounted file systems for the local system.
OBJFS	The object file system (OBJFS) describes the state of all modules currently loaded by the kernel. This file system is used by debuggers to access information about kernel symbols without accessing the kernel directly.
SHAREFS	The `sharetab` file system (SHAREFS) provides read-only access to the table of shared file systems for the local system.
PROCFS	The process file system (PROCFS) resides in memory and contains a list of active processes, by process number, in the `/proc` directory. Information in the `/proc` directory is used by commands such as `ps`. Debuggers and other development tools can also access the address space of the processes by using the PROCFS calls.
TMPFS	The temporary file system (TMPFS) uses local memory for a file system to read and write. Using TMPFS improves system performance by saving the cost of reading and writing temporary files to a local disk or across the network. For example, temporary files are created when you compile a program. The OS generates disk activity or network activity logs while manipulating these files. Using TMPFS to hold these temporary files significantly increases the speed of create, delete, and data access operations. Files in TMPFS file systems are not permanent. These files are deleted when the file system is unmounted or when the system is shut down or rebooted. TMPFS is the default file system type for the `/tmp` directory in the Oracle Solaris OS. You can copy or move files into or out of the `/tmp` directory,

	as you would in a ZFS or UFS file system. The TMPFS file system uses swap space as a temporary backing store.
LOFS	The loopback file system (LOFS) enables you create a new virtual file system so that you can access files by using an alternative path name. For example, you can create a loopback mount of the root (/) directory on `/tmp/newroot`. This loopback mount and the entire file system hierarchy appears as if it is duplicated under `/tmp/newroot`, including any file systems mounted from NFS servers. All files are accessible either with a path name starting from root (/), or with a path name that starts from `/tmp/newroot`.

Extended File Attributes

The ZFS, UFS, NFS, and TMPFS file systems are enhanced to include extended file attributes. Extended file attributes enable application developers to associate specific attributes to a file. For example, a developer of an application used to manage a windowing system might choose to associate a display icon with a file. Extended file attributes are logically represented as files within a hidden directory that are associated with the target file.

You can use the `runat` command to add attributes and execute shell commands in the extended attribute namespace. This namespace is a hidden attribute directory that is associated with a specified file.

To add attributes to a file, you must create the attributes file using the `runat` command.

```
$ runat filea cp /tmp/attrdata attr.1
```

Then, use the `runat` command to list the attributes of the file.

```
$ runat filea ls -l
```

For more information, see the `runat(1)` man page.

Many Oracle Solaris file system commands are modified to support extended file attributes, which enables you to query, copy, or find file attributes. For more information, see the specific man page for each file system command.

♦ ♦ ♦ C H A P T E R 2

Managing File Systems

This chapter describes how to create, mount, and unmount a file system. This chapter also describes how to resolve UFS file system inconsistencies and errors.

This chapter contains the following topics:

- "Creating a File System" on page 15
- "Determining the Type of a File System" on page 22
- "Mounting a File System" on page 23
- "The Virtual File System Table" on page 26
- "Unmounting a File System" on page 31
- "Checking UFS File System Consistencies" on page 32
- "Resolving UFS File System Inconsistencies" on page 36

Creating a File System

To create a new file system, you must format and label a disk or a storage device such as USB, flash disk, or memory card. You can also create a file system spanning a whole device. Usually volumes are provided by a volume manager such as Solaris volume manager (SVM). There are no procedures to create network-based and virtual file systems.

For information about how to create a ZFS file system, see *Managing ZFS File Systems in Oracle Solaris 11.3*.

Caution - While creating a new file system, ensure that you specify the correct device name. If you specify an incorrect device name, the content in that device will be erased. This error might cause the system to panic.

You can use the `mkfs` command to create a file system.

```
# mkfs -F type [-o parameters] raw_device [parameters]
```

Parameters differ based on the type of file system you want to create. The syntax can also differ from one file system to the other, though there are some common parameters such as `block-size`. For more information, see the `mkfs`(1M) man page.

Creating a UFS File System

The `mkfs` tool requires a number of parameters related to device geometry for UFS to be entered. The `newfs` tool obtains these parameters automatically..

```
# newfs [-N] [-b size] [-i bytes] /dev/rdsk/device-name
```

The `-b` option specifies the size of a block (a unit in which the data is transferred to and from the device). The default size of a block is 8192 bytes.

The `-i` option specifies the number of bytes per inode, which specifies the density of inodes in the file system. The value should reflect the expected average size of files in the file system. If you require fewer inodes, larger number must be used. To create more inodes, you must provide a smaller number.

When you issue the `newfs` command, it prompts for confirmation if the file system must be created on a given device. The `newfs` tool then invokes the `mkfs` tool with appropriate arguments for the device.

EXAMPLE 1 Creating a UFS File System

The following example shows how to create a UFS file system on `/dev/rdsk/c0t1d0s0`.

```
# newfs /dev/rdsk/c0t1d0s0
newfs: construct a new file system /dev/rdsk/c0t1d0s0: (y/n)? y
/dev/rdsk/c0t1d0s0:     286722656 sectors in 46668 cylinders of 48 tracks, 128 sectors
140001.3MB in 2917 cyl groups (16 c/g, 48.00MB/g, 5824 i/g)
super-block backups (for fsck -F ufs -o b=#) at:
32, 98464, 196896, 295328, 393760, 492192, 590624, 689056, 787488, 885920,
Initializing cylinder groups:
.........................................................
super-block backups for last 10 cylinder groups at:
285773216, 285871648, 285970080, 286068512, 286166944, 286265376, 286363808,
286462240, 286560672, 286659104
```

For more information, see the `newfs`(1M) man page.

Creating a Multi-Terabyte UFS File System

The UFS file system size is limited to 2 TB. In some environments such as a database storage, you need to store more data in very few files. UFS provides a configuration called Multi-TeraByte UFS (or MTBUFS), which can store upto 16 TB data.

To create a multi-terabyte UFS, run the `newfs` command with the `-T` option.

```
# newfs -T raw_device
```

Assuming that the file system might contain only a small number of files, the space needed for metadata (information about the files) is reduced and reused for additional data capacity.

EXAMPLE 2 Creating a Multi-Terabyte UFS File System

The following example shows how to create a MTBUFS file system on `/dev/rdsk/c2t2d0s0`.

```
# newfs -T /dev/rdsk/c2t2d0s0
newfs: construct a new file system /dev/rdsk/c2t2d0s0: (y/n)? y
Warning: 2864 sector(s) in last cylinder unallocated
/dev/rdsk/c2t2d0s0:     143299792 sectors in 23324 cylinders of 48 tracks, 128 sectors
        69970.6MB in 1458 cyl groups (16 c/g, 48.00MB/g, 64 i/g)
super-block backups (for fsck -F ufs -o b=#) at:
 32, 98464, 196896, 295328, 393760, 492192, 590624, 689056, 787488, 885920,
Initializing cylinder groups:
............................
super-block backups for last 10 cylinder groups at:
 142349344, 142447776, 142546208, 142644640, 142743072, 142841504, 142939936,
 143038368, 143136800, 143235232
```

Customizing UFS File System Parameters

Ensure that you understand the available parameters for the `newfs` command before you modify the default file system parameters. For more information about customizing these parameters, see the `newfs`(1M) and `mkfs_ufs`(1M) man pages.

Logical Block Size

The logical block size parameter can be set by using the `-b bsize` parameter with the `newfs` command.

The logical block size is the size of the block that the UNIX kernel uses to read or write a file. The logical block size is usually different from the physical block size. The physical block size is usually 512 bytes, which is the size of the smallest block that a disk controller can read or write. Logical block size is set to the page size of the system by default. The default logical block size is 8192 bytes (8 KB) for UFS file systems. The UFS file system supports block sizes of 4096 or 8192 bytes (4 or 8 KB). It is a best practice to set the logical block size to 8 KB.

To choose the best logical block size suitable for your system, you must consider both performance and available space. For most UFS systems, an 8 KB file system provides the best performance, offering a good balance between disk performance, and the use of space in the primary memory and on the disk.

To increase efficiency, you can use a larger logical block size for file systems when most of the files are large. Use a smaller logical block size for file systems when most of the files are small. You can use the `-c` option with the `newfs` command on a file system to display a complete report on the distribution of files by block size. However, the page size that is set when the file system is created is probably the best size in most cases.

Fragment Size

The fragment size parameter can be set using the `-f fragsize` parameter with the `newfs` command.

As files are created or expanded, they are allocated with disk space with either full logical blocks or portions of logical blocks called fragments. When disk space is needed for a file, full blocks are allocated first, and then one or more fragments of a block are allocated for the remainder. For small files, allocation begins with fragments.

The ability to allocate fragments of blocks to files saves space by reducing fragmentation of disk space that result from unused holes in blocks.

You define the fragment size when you create a UFS file system. The default fragment size is 1 KB. Each block can be divided into 1, 2, 4, or 8 fragments, which results in fragment sizes from 8192 bytes to 512 bytes (for 4 KB file systems only). The lower bound is actually tied to the disk sector size, typically 512 bytes.

For multi-terabyte file systems, the fragment size must be equal to the file system block size.

When choosing a fragment size, consider the trade-off between space and time. A small fragment size saves space, but requires more time to allocate. To increase storage efficiency, use a larger fragment size for file systems when most of the files are large. Use a smaller fragment size for file systems when most of the files are small.

Minimum Free Space

The minimum free space is the percentage of the total disk space that is held in reserve when you create a file system. The default reserve is ((64 MB/partition size) * 100), rounded down to the nearest integer and limited between 1 percent and 10 percent, inclusively.

Free space is important because file access becomes less efficient when a file system becomes full. As long as an adequate amount of free space exists, UFS file systems operate efficiently. When a file system becomes full, using the available user space, only `root` can access the reserved free space.

Commands such as `df` reports the percentage of space that is available to users, excluding the percentage allocated as the minimum free space. When the command reports that more than 100 percent of the disk space in the file system is in use, some of the reserve space has been used by `root`.

If you impose quotas on users, the amount of space available to them does not include the reserved free space. You can change the value of the minimum free space for an existing file system using the `tunefs` command.

Optimization Type

The optimization type (`-o` option) is set to either `space` or `time`.

`space`	When you select `space` optimization, disk blocks are allocated to minimize fragmentation and disk use is optimized.
`time`	When you select `time` optimization, disk blocks are allocated as quickly as possible, with less emphasis on their placement. When sufficient free space exists, allocating disk blocks is relatively easy, without resulting in too much fragmentation. The default is `time`.

You can change the value of the optimization type parameter for an existing file system by using the `tunefs` command. For more information, see the `tunefs`(1M) man page.

Number of Inodes

The number of bytes per inode specifies the density of inodes in the file system. The number of bytes is divided into the total size of the file system to determine the number of inodes to create. Once the inodes are allocated, you cannot change the number without re-creating the file system.

If the file system is less than 1 GB, the default number of bytes per inode is 2048 bytes (2 KB). The following table shows the number of bytes per inode for particular file system size:

File System Size	Number of Bytes Per Inode
Less than or equal to 1 GB	2048
Less than 2 GB	4096
Less than 3 GB	6144
3 GB up to 1 TB	8192
Greater than 1 TB or created with the `-T` option	1048576

If you have a file system with many symbolic links, they can lower the average file size. If the file system has many small files, provide a lower value for the inode parameter.

Caution - If you have less number of inodes, you reach the maximum number of files on a disk slice that is practically empty. Hence, ensure that the number of inodes match your requirements.

Maximum UFS File and File System Size

The maximum size of a UFS file system is about 16 TB of usable space, minus about one percent overhead. A sparse file can have a logical size of one terabyte. However, the actual amount of data that can be stored in a file is approximately one percent less than 1 TB because of the file system overhead.

Maximum Number of UFS Subdirectories

The maximum number of subdirectories per directory in a UFS file system is 32,767. This limit is predefined and cannot be changed.

Creating a PCFS File System

To create a PCFS (or FAT) file system, use the `mkfs` command. However, PCFS requires special care for specifying the device and might require some additional parameters. The syntax for the `mkfs` command is as follows:

```
# mkfs -F pcfs [-o parameters] device-name[:logical-name] [size]
```

As PCFS is usually placed on a DOS partition, Oracle Solaris uses a special naming scheme to access these partitions. The device specification consists of two parts:

`/dev/rdsk/device-name`	Specifies the device name of the whole disk. For example, `/dev/rdsk/c0t0d0p0` on an x86 system or `/dev/rdsk/c0t0d0s2` on a SPARC system.
`logical-drive`	Specifies either the DOS logical drive letter (`C` through `Z`) or a drive number (1 through 24). Drive `C` is equivalent to drive 1 and represents the primary DOS partition on the drive. All other letters or numbers represent DOS logical drives within the extended DOS partition.

Alternatively, the partition might be specified as `/dev/rdsk/c0t0d0pX` on an x86 system, where `X` is 1 to 4 for primary partitions and 5 to 32 for logical partitions. However, `mkfs` requires additional argument `nofdisk`. This parameter has to also be used when the PCFS is created on a slice or on a whole device.

Caution - Ensure that the partition does not cover the sectors containing the label information. PCFS does not protect the first sectors of a partition or slice (unlike UFS and ZFS), and overwrites them with its own data. Creating PCFS file system on such a partition or slice destroys the label and all data on the device is lost.

Creating a File System for CD or DVD

There are several file systems that can be used on CD or DVD media. Oracle Solaris supports HSFS (High Sierra File System also known as ISO-9660) and UDF (Universal Disk Format) file system. `mkisofs` is pre-mastering tool used to create pure or hybrid image of HSFS/UDF/HFS file systems with optional Rock-Ridge extension. Hybrid image is an effective combination of selected file systems containing information for all of them. Such an image can be accessed as any of the reselected ones. The syntax for the `mkisofs` command is as follows:

```
# mkisofs [options] [-o image-name] path [path ...]
```

`mkisofs` creates an image containing all files specified by paths, and stores it to a file specified by *image-name* (or the standard output if none is given). The image can then be burned to a CD or DVD media using `cdrecord` or a similar tool.

As `mkisofs` is a third party tool, certain combinations of parameters might result in an unreadable image by Oracle Solaris.

EXAMPLE 3 Creating an HSFS Image Using the `mkisofs` Command

The following example shows how to create an HSFS compliant image with Rock-Ridge extension using the `mkisofs` command. The content is specified by a path to a directory.

```
# mkisofs -R -J -o /data/hsfsimage.iso /data/content
  0.12% done, estimate finish Mon Sep 12 15:01:35 2016
  0.24% done, estimate finish Mon Sep 12 14:47:33 2016
...
 49.97% done, estimate finish Mon Sep 12 14:43:05 2016
 50.09% done, estimate finish Mon Sep 12 14:43:04 2016
 50.21% done, estimate finish Mon Sep 12 14:43:04 2016
...
 99.82% done, estimate finish Mon Sep 12 14:43:19 2016
 99.94% done, estimate finish Mon Sep 12 14:43:19 2016
Total translation table size: 0
Total rockridge attributes bytes: 2175
Total directory bytes: 0
Path table size(bytes): 10
Max brk space used 14000
4222681 extents written (8247 MB)
```

Determining the Type of a File System

If you have a raw device name of a disk slice, you can use the `fstyp` or the `df` command to determine the type of a file system. The `fstyp` command might not be able to identify third party file systems, which are not supported by Oracle Solaris.

For more information, see the `fstyp`(1M) and `df`(1M) man pages.

EXAMPLE 4 How to Determine the Type of a File System

The following example uses the `df -n` command to display the mounted file systems along with its type. The `df` command uses information about mounted file systems provided by the kernel.

```
# df -n
/                   : zfs
/devices            : devfs
/dev                : dev
/system/contract    : ctfs
/proc               : proc
/etc/mnttab         : mntfs
/system/volatile    : tmpfs
```

```
/system/object      : objfs
/etc/dfs/sharetab   : sharefs
/dev/fd             : fd
/var                : zfs
/tmp                : tmpfs
/var/share          : zfs
/export             : zfs
/export/home        : zfs
/rpool              : zfs
/media/cdrom        : ufs
/media/cdrom-1      : ufs
/media/cdrom-2      : ufs
/media/cdrom-3      : ufs
/media/sol_10_811_sparc : hsfs
/media/cdrom-4      : ufs
/pond               : zfs
/pond/amy           : zfs
/pond/dr            : zfs
/pond/rory          : zfs
```

The following example uses the `fstyp` command to determine the file system type. The specified device might or might not be mounted.

```
# fstyp /dev/rdsk/c0t0d0s0
zfs
```

Mounting a File System

Mounting operation connects the file system subtree that is on a specific device to the file system tree. The transition between different file systems is transparent and is equal to navigating to another directory.

To mount a file system, use the `mount` command. For more information, see the `mount`(1M) man page.

For information about mounting ZFS file systems, see "Mounting ZFS File Systems" in *Managing ZFS File Systems in Oracle Solaris 11.3*.

Mounting a Disk Based File System

Apart from the device specification, all the command arguments are similar to other file systems. For disk based file system, a path to block device specifies the device. The syntax for the `mount` command is as follows:

```
# mount [-F fs-type] [-o mount-options] /dev/dsk/device-name /directory-name
```

`-F` *fs-type*	The type of file system to be mounted. If the option is not set, `mount` assumes that it is a UFS file system.
`-o` *mount-options*	Some mount options are common to all types of file system. However, each file system provides a different list of mount options depending on its features. Some of the common options are `rw` and `ro`, which set the access to the file system to read and write, or read-only.
/dev/dsk/*device-name*	The path to the block device of a slice or partition containing the file system.
/*directory-name*	The directory where you want to mount the file system.

When a file system is specified in the virtual file system table `/etc/vfstab`, you can omit the device name or the mount point. The `mount` command refers to the `/etc/vfstab` file for missing information and mount options, and merges them with the arguments provided by the user.

EXAMPLE 5 Mounting a UFS File System

The following example shows how to mount a UFS in slice 3 to the `/export/park` directory.

```
# mount -F ufs /dev/dsk/c0t0d0s3 /export/park
```

Mounting a PCFS File System

Oracle Solaris uses special naming scheme to access DOS partitions where PCFS file system is usually stored. The resource consists of a path to a block device of the whole disk and a logical drive specifier.

```
# mount -F pcfs /dev/dsk/device-name:logical-drive /mount-point
```

/dev/dsk/*device-name*	Specifies the device name of the whole disk. For example, `/dev/dsk/c0t0d0p0` on an x86 system or `/dev/dsk/c0t0d0s2` on a SPARC system.
logical-drive	Specifies either the DOS logical drive letter (C through Z) or a drive number (1 through 24). Drive C is equivalent to drive 1, and represents the primary DOS partition on the drive. All other letters or numbers represent DOS logical drives within the extended DOS partition.

Note - You must use a colon to separate the *device-name* and *logical-drive*.

EXAMPLE 6 Mounting a PCFS File System From a Primary Partition on x86 and SPARC Systems

The following example shows how to mount a logical drive in the primary DOS partition on the `/pcfs/c` directory on an x86 system.

```
# mount -F pcfs /dev/dsk/c0t0d0p0:c /pcfs/c
```

The following example shows how to mount a logical drive in the primary DOS partition on the `/pcfs/c` directory on a SPARC system.

```
# mount -F pcfs /dev/dsk/c0t0d0s2:c /pcfs/c
```

EXAMPLE 7 Mounting a PCFS (DOS) File System From a Logical Partition on x86 and SPARC Systems

The following example shows how to read only mount the first logical drive in the extended DOS partition located on the `/mnt` directory on an x86 system.

```
# mount -F pcfs -o ro /dev/dsk/c0t0d0p0:2 /mnt
```

The following example shows how to read only mount the first logical drive in the extended DOS partition located on the `/mnt` directory on a SPARC system.

```
# mount -F pcfs -o ro /dev/dsk/c0t0d0s2:2 /mnt
```

Caution - On an x86 system, the DOS partitions are available as devices named `c0t0d0p1` to `c0t0d0p4` for primary partitions and `c0t0d0p5` to `c0t0d0p32` for logical partitions. Those can also be used to mount and access PCFS file system. Ensure to use notation either consistently or exclusively. Using the same file system twice can corrupt the file system.

Mounting a Network Based File System

For a network based file system, the resource specification consists of a server name and a path to a directory. This resource specification is also called a share. For more information about how to create a share over a specific network protocol see, *Managing Network File Systems in Oracle Solaris 11.3* and *Managing SMB File Sharing and Windows Interoperability in Oracle Solaris 11.3*. You can also refer to your operating system documentation.

```
# mount -F (nfs|smbfs) [-o mount-options] server:/directory /mount-point
```

You can automate the network based file systems by using the `automount` command. For more information about the `automounter`, see Chapter 4, "Administering Autofs" in *Managing Network File Systems in Oracle Solaris 11.3* and the `automount`(1M) man page.

EXAMPLE 8 Mounting an NFS File System

The following example shows how to mount the `/export/packages` directory on `/mnt` from the server, `pluto`.

```
# mount -F nfs pluto:/export/packages /mnt
```

EXAMPLE 9 Mounting a CIFS or an SMB File System

The following example shows how to mount the `/data/workspace` directory on `/mnt` from the server, `neptune`.

```
# mount -F smbfs neptune:/data/workspace /mnt
```

Mounting a File System Image

You can specify a regular file containing a file system image instead of specifying a device.

```
# mount -F fs-type [-o mount-options] path-to-image-file /mount-point
```

Depending on the capability of the file system driver, an image is mounted as read-only, or read-write. For example, HSFS image can be mounted only for reading while PCFS image can be mounted for reading or writing.

EXAMPLE 10 Mounting an Image of HSFS File System

The following example shows how to mount the HSFS image stored in `/tank/build/image.iso` on the `/mnt` file system.

```
# mount -F hsfs /tank/build/image.iso /mnt
```

The Virtual File System Table

Most file systems are mounted automatically by an SMF service at the system boot time. All file systems except for ZFS are mounted according to the `/etc/vfstab` file. The list of ZFS file systems to mount is taken from ZFS cache.

To add an entry for mounting a legacy or remote file system, the information you need to specify are as follows:

- The device or the server where the file system resides

- The file system mount point
- File system type
- Whether you want the file system to mount automatically when the system boots
- Any mount options

The following `vfstab` example is from a system that has a ZFS root file system. In addition, this system is mounting a remote file system, `/users/data`, from the NFS server, `neo`.

```
# cat /etc/vfstab
#device         device          mount           FS      fsck    mount   mount
#to mount       to fsck         point           type    pass    at boot options
#
fd              -                /dev/fd        fd      -       no      -
/proc           -                /proc          proc    -       no      -
/dev/zvol/dsk/rpool/swap -       -              swap    -       no      -
/devices        -               /devices        devfs   -       no      -
sharefs         -               /etc/dfs/sharetabsharefs -      no      -
ctfs            -               /system/contract ctfs   -       no      -
objfs           -               /system/object  objfs   -       no      -
swap            -               /tmp            tmpfs   -       yes     -
neo:/users/data -               /data           nfs     -       yes     -
```

You can mount ZFS file systems from the `vfstab` file by using the legacy mount feature.

Field Descriptions for the `/etc/vfstab` File

An entry in the `/etc/vfstab` file has seven fields, which are described in the following list.

`device to mount` This field identifies one of the following resources:

- Block device name for a local legacy UFS file system. For example, `/dev/dsk/c8t1d0s7`.
- Resource name for a remote file system. For example, `myserver:/export/home`.

  After you add an entry for a remote system resource, ensure that the following service is enabled.

  ```
  # svcs -a | grep nfs/client
  disabled       May_14   svc:/network/nfs/client:default
  # svcadm enable svc:/network/nfs/client:default
  ```
- Swap volume. For example, `/dev/zvol/dsk/rpool/swap`.
- Directory for a virtual file system.

`device to fsck`	Determines the raw interface that is used by the `fsck` command. Use a dash (-) when there is no applicable device, such as for a read-only file system or a remote file system. The raw device name that corresponds to the legacy UFS file system identified by the device to mount field.
`mount point`	Identifies where to mount the legacy or remote file system.
`FS type`	Identifies the type of file system.
`fsck pass`	The pass number indicates whether to check a legacy UFS file system. When the field contains a dash (-), the file system is not checked. When this field is zero, legacy UFS file systems are not checked. When the field value is greater than zero, the UFS file system is checked. All legacy UFS file systems with a value of 1 in this field are checked one at a time in the order they appear in the `vfstab` file. When the `fsck` command is run on multiple UFS file systems that have `fsck pass` values greater than 1, using the `preen` option, the `fsck` command automatically checks the file systems on different disks in parallel to maximize efficiency. Otherwise, the value of the pass number does not have any effect.
`mount at boot`	Specifies whether the file system must be automatically mounted by `mountall` command when the system is booted. The option can be `yes` or `no`. This field has nothing to do with `autofs`. This field should always be set to `no` for virtual file systems such as `/proc` and `/dev/fd`.
`mount options`	A list of comma separated options (with no spaces) that are used for mounting the file system. Use a dash (-) to indicate no options. For more information, see the `vfstab`(4) man page.

Note - You must have an entry in each field in the `/etc/vfstab` file. If there is no value for a field, the system boot might not be successful. Ensure that you add a dash (-) because white space cannot be used as a field value.

For more information about NFS, see *Managing Network File Systems in Oracle Solaris 11.3*.

Displaying the Mounted File System

You can display the list of mounted file systems. There are several tools to display various information about all the mounted file systems. For example, the `df` command displays the capacity and free space on each file system.

For more information about the mounted file system, you can either check the output of the `mount` command or check the content of the `/etc/mnttab` file.

Using the Mounted File System Table

When you mount or unmount a file system, the `/etc/mnttab` file is updated with the list of currently mounted file systems. This file cannot be edited by any user including the administrator. The following example shows the `/etc/mnttab` file.

```
$ more /etc/mnttab
rpool/ROOT/zfsBE    /       zfs     dev=3390002     0
/devices        /devices        devfs   dev=8580000     1337114941
/dev    /dev    dev     dev=85c0000     1337114941
ctfs    /system/contract        ctfs    dev=8680001     1337114941
proc    /proc   proc    dev=8600000     1337114941
mnttab  /etc/mnttab     mntfs   dev=86c0001     1337114941
swap    /system/volatile        tmpfs   xattr,dev=8700001   1337114941
objfs   /system/object  objfs   dev=8740001     1337114941
sharefs /etc/dfs/sharetab       sharefs dev=8780001     1337114941
/usr/lib/libc/libc_hwcap2.so.1  /lib/libc.so.1  lofs dev=3390002  13371149
fd      /dev/fd fd      rw,dev=8880001  1337114969
rpool/ROOT/zfsBE/var        /var    zfs
 rw,devices,setuid,nonbmand,exec,rstchown,xattr,atime,dev=3390003        1337114969
swap    /tmp    tmpfs   xattr,dev=8700002       1337114969
rpool/VARSHARE  /var/share      zfs
 rw,devices,setuid,nonbmand,exec,rstchown,xattr,atime,dev=3390004        133711496
```

Obtaining the List of Mounted File Systems Using the `mount` Command

In addition to mounting file systems, the `mount` command displays the list of all the mounted file systems when resource and mount point are not specified. This list can be altered by using the `-p` option to obtain a format more suitable for processing by other programs or scripts.

The following example shows the list of mounted file systems using the `mount` command without any arguments.

```
# mount
/ on rpool/ROOT/zfsBE read/write/setuid/devices/rstchown/dev=45d0002 on Thu Jan  1
 01:00:00 1970
/devices on /devices read/write/setuid/devices/rstchown/dev=8b00000 on Mon Jun 20
 14:46:48 2016
```

```
/dev on /dev read/write/setuid/devices/rstchown/dev=8b40000 on Mon Jun 20 14:46:48 2016
/system/contract on ctfs read/write/setuid/devices/rstchown/dev=8c40001 on Mon Jun 20
 14:46:48 2016
/proc on proc read/write/setuid/devices/rstchown/dev=8b80000 on Mon Jun 20 14:46:48 2016
/etc/mnttab on mnttab read/write/setuid/devices/rstchown/dev=8c80001 on Mon Jun 20
 14:46:48 2016
/system/volatile on swap read/write/setuid/devices/rstchown/xattr/dev=8cc0001 on Mon Jun
 20 14:46:48 2016
/system/object on objfs read/write/setuid/devices/rstchown/dev=8d00001 on Mon Jun 20
 14:46:48 2016
/etc/dfs/sharetab on sharefs read/write/setuid/devices/rstchown/dev=8d40001 on Mon Jun
 20 14:46:48 2016
/lib/libc.so.1 on /usr/lib/libc/libc_hwcap2.so.1 read/write/setuid/devices/rstchown/
dev=45d0002 on Mon Jun 20 14:47:03 2016
/dev/fd on fd read/write/setuid/devices/rstchown/dev=8e00001 on Mon Jun 20 14:47:04 2016
/var on rpool/ROOT/s12_101/var read/write/setuid/devices/rstchown/nonbmand/exec/xattr/
atime/dev=45d0004 on Mon Jun 20 14:47:36 2016
/tmp on swap read/write/setuid/devices/rstchown/xattr/dev=8cc0002 on Mon Jun 20 14:47:36
 2016
/var/share on rpool/VARSHARE read/write/nosetuid/devices/rstchown/nonbmand/noexec/
noxattr/atime/dev=45d0005 on Mon Jun 20 14:47:37 2016
/export on rpool/export read/write/setuid/devices/rstchown/nonbmand/exec/xattr/atime/
dev=45d0006 on Mon Jun 20 14:47:53 2016
/export/home on rpool/export/home read/write/setuid/devices/rstchown/nonbmand/exec/
xattr/atime/dev=45d0007 on Mon Jun 20 14:47:54 2016
/export/home/jack on rpool/export/home/jack read/write/setuid/devices/rstchown/nonbmand/
exec/xattr/atime/dev=45d0008 on Mon Jun 20 14:47:56 2016
/rpool on rpool read/write/setuid/devices/rstchown/nonbmand/exec/xattr/atime/dev=45d0009
 on Mon Jun 20 14:47:56 2016
/mnt on /dev/dsk/c2t1d0s0 read/write/setuid/devices/rstchown/intr/largefiles/logging/
xattr/onerror=panic/dev=3400080 on Tue Jun 21 13:39:25 2016
```

The following example shows the output suitable for automated parsing. The format is the same as the `/etc/vfstab` file.

```
# mount -p
rpool/ROOT/zfsBE - / zfs - no
/devices - /devices devfs - no
/dev - /dev dev - no
ctfs - /system/contract ctfs - no
proc - /proc proc - no
mnttab - /etc/mnttab mntfs - no
swap - /system/volatile tmpfs - no xattr
objfs - /system/object objfs - no
sharefs - /etc/dfs/sharetab sharefs - no
/usr/lib/libc/libc_hwcap2.so.1 - /lib/libc.so.1 lofs - no
fd - /dev/fd fd - no rw
```

```
rpool/ROOT/zfsBE/var - /var zfs - no
 rw,devices,setuid,nonbmand,exec,rstchown,xattr,atime
swap - /tmp tmpfs - no xattr
rpool/VARSHARE - /var/share zfs - no
 rw,devices,nosetuid,nonbmand,noexec,rstchown,noxattr,atime
rpool/export - /export zfs - no rw,devices,setuid,nonbmand,exec,rstchown,xattr,atime
rpool/export/home - /export/home zfs - no
 rw,devices,setuid,nonbmand,exec,rstchown,xattr,atime
rpool/export/home/jack - /export/home/jack zfs - no
 rw,devices,setuid,nonbmand,exec,rstchown,xattr,atime
rpool - /rpool zfs - no rw,devices,setuid,nonbmand,exec,rstchown,xattr,atime
/dev/dsk/c2t1d0s0 - /mnt ufs - no rw,intr,largefiles,logging,xattr,onerror=panic
```

Unmounting a File System

You can use the `unmount` command to unmount a file system. For more information about unmounting a ZFS file system, see "Unmounting ZFS File Systems" in *Managing ZFS File Systems in Oracle Solaris 11.3*.

You can either unmount a specific file system or unmount a specific device that contains a file system. For more information, see the `umount`(1M) man page.

Ensure that the file system is not busy. A file system is considered busy, if a user is accessing a directory in the file system, or if a program has an open file in that file system, or if the file system is being shared.

Perform the following operations to make a file system available for unmounting:

- Change to a directory in a different file system.
- Stop the processes from using files on the file system.
- Unshare the file system. For more information about unsharing a file system, see the `unshare`(1M) man page.

To verify that you have unmounted a file system, use the `mount` command.

```
$ mount | grep unmounted-file-system
```

▼ How to Stop All Processes That Are Accessing a File System

1. **Become an administrator.**

For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.3*.

2. **List all the processes that are accessing the file system to know the processes that you want to stop.**

   ```
   # fuser -c [ -u ] /mount-point
   ```

`-c`	Reports on files that are mount points for file systems and any files within those mounted file systems
`-u`	Displays the user login name for each process ID
/mount-point	Specifies the name of the file system for which you want to stop processes

3. **Stop all processes that are accessing the file system.**

   ```
   # fuser -c -k /mount-point
   ```

   A `SIGKILL` is sent to each process that is using the file system.

   Note - You must not stop the processes of a user without first warning the user.

4. **Verify that no process is accessing the file system.**

   ```
   # fuser -c /mount-point
   ```

Example 11 Stopping All Processes That Are Accessing a File System

This example shows how to stop process `4006c` that is using the `/export/home` file system.

```
# fuser -c /export/home
/export/home:     4006c
# fuser -c -k /export/home
/export/home:     4006c
# fuser -c /export/home
/export/home:
```

Checking UFS File System Consistencies

You might need to interactively check a file system when you are unable to mount it or when the file system develops inconsistencies.

When a file system in use develops inconsistencies, error messages are displayed on the console window or on the system message file, or the system might crash. For example, the system messages file, `/var/adm/messages`, might include messages such as:

```
Sep  5 13:42:40 hostname ufs: [ID 879645 kern.notice] NOTICE: /: unexpected
free inode 630916, run fsck(1M)
```

hostname is the system which is reporting the error.

Consider the following aspects when running the `fsck` command to check UFS file systems:

- A file system must be inactive in order to be checked. File system changes waiting to be flushed to disk or file system changes that occur during `fsck` checking process can be interpreted as file system corruption. These issues might not be a reliable indication of a problem.
- A file system must be inactive in order to be repaired. File system changes waiting to be flushed to disk or file system changes that occur during the `fsck` repairing process might cause the file system corruption. This might also cause the system to crash.
- Unmount a file system before you use `fsck` on that file system. Unmounting the file system ensures the data structures in it are consistent.

If a UFS file system runs out of space, you can see the following message in the `/var/adm/messages` file.

```
file system full
```

For more information about troubleshooting file system issues, see Chapter 3, "Troubleshooting File System Problems" in *Troubleshooting System Administration Issues in Oracle Solaris 11.3*.

▼ How to Check a UFS File System

There is no need to rerun the `fsck` command if you see the following message:

```
***** FILE SYSTEM WAS MODIFIED *****
```

This message is an informational message about the actions of `fsck`. You can run `fsck` after the message is displayed.

1. **Become an administrator.**

   For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.3*.

2. **Unmount the local file system to ensure that there is no activity on the file system.**

   Specify the mount point directory or `/dev/dsk/`*device-name* as arguments to the `fsck` command. If there are any inconsistencies, error messages are displayed.

   ```
   # umount /export/home
   # fsck /dev/rdsk/c0t0d0s7
   ** /dev/dsk/c0t0d0s7
   ** Last Mounted on /export/home
   .
   .
   ```

3. **Correct any reported `fsck` errors.**

   For information about how to respond to the error message prompts while you check one or more UFS file systems, see "Resolving UFS File System Inconsistencies" on page 36.

4. **Mount the repaired file system to determine if there are any files in the `lost+found` directory.**

   The `fsck` command moves individual files to the `lost+found` directory and these files are renamed with their inode numbers.

5. **Rename and move any files in the `lost+found` directory.**

   Rename the files and move them to their original location. Use the `grep` command to match phrases within individual files and the `file` command to identify the type of file.

   Remove all the unidentifiable files or directories left in the `lost+found` directory so that the directory does not fill.

   If `fsck` cannot repair all of the problems, see "Fixing a UFS File System That the `fsck` Command Cannot Repair" on page 57.

Example 12 Interactive Checking of UFS File Systems

The following example shows how to check the `/dev/rdsk/c0t0d0s6` file system and correct the incorrect block count. This example assumes that the file system is unmounted.

```
# fsck /dev/rdsk/c0t0d0s6
** Phase 1 - Check Block and Sizes
INCORRECT BLOCK COUNT I=2529 (6 should be 2)
CORRECT? y

** Phase 2 - Check Pathnames
```

```
** Phase 3 - Check Connectivity
** Phase 4 - Check Reference Counts
** Phase 5 - Cylinder Groups
929 files, 8928 used, 2851 free (75 frags, 347 blocks, 0.6%
fragmentation)

***** FILE SYSTEM WAS MODIFIED ****
```

Preening UFS File Systems

The `fsck -o p` command checks a UFS file system and automatically fix the problems that normally result from an unexpected system shutdown. This command exits immediately if it encounters a problem that requires operator intervention. This command also permits parallel checking of the file systems.

You can run the `fsck -o p` command to preen the file systems after an unclean shutdown. In this mode, the `fsck` command does not look at the clean flag and performs a full check. These actions are a subset of the actions performed by the interactive `fsck` command.

▼ How to Preen a UFS File System

1. **Become an administrator.**

   For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.3*.

2. **Unmount the UFS file system.**

   ```
   # umount /mount-point
   ```

3. **Check the UFS file system using the preen (-p) option.**

   ```
   # fsck -o p /dev/rdsk/device-name
   ```

   You can preen individual file systems by using /*mount-point* or `/dev/rdsk/`*device-name* as arguments to the `fsck` command.

Example 13 Preening a UFS File System

The following example shows how to preen the `/export/home` file system.

```
# fsck -o p /export/home
```

Resolving UFS File System Inconsistencies

The `fsck` command is run non-interactively to preen the file systems after an abrupt system halt, where the latest file system changes are not written to disk. Preening automatically fixes any basic file system inconsistencies and does not repair serious errors. While preening a file system, the `fsck` command fixes the inconsistencies it expects from such an abrupt halt. For serious conditions, the command reports the error and terminates.

When you run the `fsck` command interactively, it reports each inconsistency found and fixes non-fatal errors. However, the command reports the inconsistency and prompts you to choose a response for serious errors. When you run the `fsck` command using the `-y` or `-n` option, your response is predefined as yes or no to the default response suggested by the `fsck` command for each error condition.

A few corrective actions might result in some loss of data. You can determine the amount and severity of data loss from the `fsck` diagnostic output.

The `fsck` command is a multipass file system check program. Each pass invokes a different phase of the `fsck` command with different set of messages. After initialization, the `fsck` command performs successive passes over each file system, checking blocks and sizes, path names, connectivity, reference counts, and the map of free blocks (possibly rebuilding it). It also performs cleanup.

The passes (phases) performed by the UFS version of the `fsck` command are as follows:

- Initialization
- Phase 1 – Check blocks and sizes
- Phase 2a – Check duplicated names
- Phase 2b – Check pathnames
- Phase 3 – Check connectivity
- Phase 3b – Verify shadows and access control lists (ACLs)
- Phase 4 – Check reference counts
- Phase 5 – Check cylinder groups

The following sections describe the error conditions that might be detected in each phase, the messages and prompts that result, and the possible responses you can provide.

Messages that might appear in more than one phase are described in “General `fsck` Error Messages” on page 37. Other messages are organized alphabetically by the phases in which they occur.

The abbreviations that appear in the `fsck` command error messages are listed as follows:

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

General `fsck` Error Messages

The error messages in this section might be displayed in any phase after initialization. Although they offer the option to continue, it is a best practice to consider them as fatal. They reflect a serious system failure and must be handled immediately. When confronted with such messages, terminate the program by entering `n` (no). If you are unable to determine the cause for the problem, contact your local service provider.

`CANNOT SEEK: BLK` *block-number* `(CONTINUE)`

Cause: A request to move to the specified block number, in the file system failed. This message indicates a serious hardware failure. If you want to continue the file system check, `fsck` retries to move the specified block number and displays a list of sector numbers that cannot be moved. If the block is a part of the virtual memory buffer cache, `fsck` terminates with a fatal I/O error message.

Solution: If the disk is experiencing hardware problems, the problem will persist. Run the `fsck` command again to recheck the file system. If the recheck fails, contact your local service provider.

`CANNOT READ: DISK BLOCK` *block-number*`: I/O ERROR`
`CONTINUE?`

Cause: A request to read a specified block number in the file system failed. The message indicates a serious problem such as hardware failure. If you want to continue the file system check, `fsck` retries to read the specified block number and displays a list of sector numbers that cannot be read. If the block is a part of the virtual memory buffer cache, `fsck` terminates with a fatal I/O error message. If `fsck` tries to write back one of the blocks on which the read failed, it displays the following message:

`WRITING ZERO'ED BLOCK` *sector-numbers* `TO DISK`

Solution: If the disk is experiencing hardware problems, the problem will persist. Run `fsck` again to check the file system. If the check fails, contact your local service provider.

`CANNOT WRITE: BLK` *block-number* `(CONTINUE)`

Cause: A request to write a specified block number in the file system failed. If you continue the file system check, `fsck` retries to write the specified block number and displays a list of sector numbers that cannot be written. If the block is a part of the virtual memory buffer cache, `fsck` terminates with a fatal I/O error message.

Solution: The disk might be write-protected. Check the write-protect lock on the drive. If the disk has hardware problems, the problem will persist. Run `fsck` again to check the file system. If write-protect is not the problem or if the check fails, contact your local service provider.

Initialization Phase `fsck` Messages

In the initialization phase, command-line syntax is checked. Before the file system check can be performed, the `fsck` command sets up tables and opens files.

The messages in this section relate to error conditions resulting from command-line options, memory requests, opening of files, status of files, file system size checks, and the creation of the scratch file. All such initialization errors terminate `fsck` when it is preening the file system.

```
Can't roll the log for device-name.
```

```
DISCARDING THE LOG MAY DISCARD PENDING TRANSACTIONS.
DISCARD THE LOG AND CONTINUE?
```

Cause: The `fsck` command was unable to flush the transaction log of a logging UFS file system prior to checking the file system for errors.

Solution: If you answer `yes` and continue, the file system operations that are in the log, but have not been applied to the file system, are lost. In this case, `fsck` runs the same checks it always runs and asks the following question in phase 5:

```
FREE BLK COUNT(S) WRONG IN SUPERBLK (SALVAGE)
```

Answering `yes` reclaims the blocks that were used for the log. The next time the file system is mounted with logging enabled, the log is recreated. Answering `no` preserves the log and exits, but the file system is not mountable.

`bad inode number` *inode-number* `to ginode`

Cause: An internal error occurred because of a nonexistent inode. This error exits `fsck`.

Solution: Contact your local service provider.

```
cannot alloc size-of-block map bytes for blockmap
cannot alloc size-of-free map bytes for freemap
cannot alloc size-of-state map bytes for statemap
cannot alloc size-of-lncntp bytes for lncntp
```

Cause: Request for memory for its internal tables failed. This failure terminates `fsck`. The error message indicates a serious system failure that must be handled immediately. This condition might occur if other processes are using a large amount of system resources.

Solution: Killing other processes might solve the problem. If not, contact your local service provider.

`Can't open checklist file:` *filename*

Cause: The file system checklist file, (`/etc/vfstab`) cannot be opened to read. This failure terminates `fsck`.

Solution: Check if the file exists and if the access modes of the file permits read access.

`Can't open` *filename*

Cause: The `fsck` command cannot open the file system, *filename*. When `fsck` is running in an interactive mode, it ignores this file system and continues to check the next file system.

Solution: Check to see if read and write access to the raw device file for the file system is permitted.

`Can't stat root`

Cause: The `fsck` command request for statistics about the `root` directory failed. This failure terminates `fsck`.

Solution: This message indicates a serious system failure. Contact your local service provider.

`Can't stat` *filename*
`Can't make sense out of name` *filename*

Cause: The `fsck` command request for statistics about the file system, *filename* failed. When `fsck` is running in an interactive mode, it ignores this file system and continues to check the next given file system.

Solution: Check if the file system exists and check its access modes.

filename: `(NO WRITE)`

Cause: Either the `-n` option is specified or `fsck` cannot open the file system, *filename* to write. When `fsck` is running in no-write mode, all diagnostic messages are displayed, but `fsck` does not attempt to fix any errors.

Solution: If the `-n` option is not specified, check the type of the file specified. It might be the name of a regular file.

`IMPOSSIBLE MINFREE=percent IN SUPERBLOCK (SET TO DEFAULT)`

Cause: The superblock minimum space percentage is greater than 99 percent or less than 0 percent.

Solution: To set the `minfree` parameter to the default 10 percent, type `y` at the default prompt. To ignore the error condition, type `n` at the default prompt.

```
filename: BAD SUPER BLOCK: message
USE AN ALTERNATE SUPER-BLOCK TO SUPPLY NEEDED INFORMATION;
e.g., fsck[-f ufs] -o b=# [special ...]
where # is the alternate superblock.  See fsck_ufs(1M)
```

Cause: The superblock is corrupted.

Solution: One of the following messages might be displayed:

```
CPG OUT OF RANGE

FRAGS PER BLOCK OR FRAGSIZE WRONG

INODES PER GROUP OUT OF RANGE

INOPB NONSENSICAL RELATIVE TO BSIZE

MAGIC NUMBER WRONG

NCG OUT OF RANGE

NCYL IS INCONSISTENT WITH NCG*CPG

NUMBER OF DATA BLOCKS OUT OF RANGE

NUMBER OF DIRECTORIES OUT OF RANGE
```

ROTATIONAL POSITION TABLE SIZE OUT OF RANGE

SIZE OF CYLINDER GROUP SUMMARY AREA WRONG

SIZE TOO LARGE BAD VALUES IN SUPERBLOCK

Try to rerun `fsck` with an alternate superblock. Specifying block 32 is a good choice. You can locate an alternate copy of the superblock by running the `newfs -N` command on the slice. If you do not specify the `-N` option, the `newfs` command overwrites the existing file system.

UNDEFINED OPTIMIZATION IN SUPERBLOCK (SET TO DEFAULT)

Cause: The superblock optimization parameter (`OPT_TIME` or `OPT_SPACE`) is not defined.

Solution: To minimize performance time of operations on the file system, type `y` at the `SET TO DEFAULT` prompt. To ignore this error condition, type `n`.

Phase 1 – Check Blocks and Sizes Messages

This phase checks the inode list. It reports error in the following conditions:

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

All errors in this phase except `INCORRECT BLOCK COUNT`, `PARTIALLY TRUNCATED INODE`, `PARTIALLY ALLOCATED INODE`, and `UNKNOWN FILE TYPE` terminate `fsck` when it is preening a file system.

The following messages might occur in phase 1 of `fsck` check.

block-number BAD I=*inode-number*

Cause: The inode, *inode-number* contains a block number, *block-number*, with a number lower than the number of the first data block in the file system or greater than the number of the last block in the file system. This error condition might generate the `EXCESSIVE BAD BLKS` error message in phase 1, if inode has too many block numbers outside the file system range. This generates `BAD/DUP` error message in phases 2 and 4.

`BAD MODE: MAKE IT A FILE?`

Cause: The status of a given inode is set to 1s, indicating file system damage. This message does not indicate physical disk damage, unless it is displayed repeatedly after running the `fsck -y` command.

Solution: Type `y` to reinitialize the inode to a reasonable value.

`BAD STATE` *state-number* `TO BLKERR`

Cause: An internal error has scrambled the `fsck` state map so that it shows the impossible value of *state-number*. This error exits `fsck` immediately.

Solution: Contact your local service provider.

fragment-number `DUP I=`*inode-number*

Cause: The inode *inode-number*, contains a block number, *fragment-number*, which is already claimed by the same inode or another inode. If inode has too many block numbers claimed by the same inode or another inode, this error condition might generate `EXCESSIVE DUP BLKS` error message in phase 1. This error condition invokes phase 1B and generates the `BAD/DUP` error messages in phases 2 and 4.

`DUP TABLE OVERFLOW (CONTINUE)`

Cause: The `fsck` command is unable to allocate memory to track duplicate fragments. If the `-o p` option is specified, the program terminates.

Solution: To continue the program, type `y` at the `CONTINUE` prompt. When this error occurs, a complete check of the file system is not possible. If another duplicate fragment is found, this error condition repeats. Increase the amount of virtual memory available by killing some processes or increase swap space to run `fsck` again to check the file system. To terminate the program, type `n`.

`EXCESSIVE BAD FRAGMENTS I=`*inode-number* `(CONTINUE)`

Cause: Too many (more than 10) fragments indicate an invalid disk address. If the `-o p` option is specified, the program terminates.

Solution: To continue the program, type `y` at the `CONTINUE` prompt. When this error occurs, a complete check of the file system is not possible. Run `fsck` again to check the file system. To terminate the program, type `n`.

`EXCESSIVE DUP BLKS I=`*inode-number* `(CONTINUE)`

Cause: Too many (more than 10) fragments are claimed by the same inode, or another inode, or by a free-list. If the `-o p` option is specified, the program terminates.

Solution: To continue the program, type `y` at the `CONTINUE` prompt. When this error occurs, a complete check of the file system is not possible. Run `fsck` again to check the file system. To terminate the program, type `n`.

`INCORRECT BLOCK COUNT I=`*inode-number* `(number-of-BAD-DUP-or-missing-blocks`
`should be number-of-blocks-in-filesystem) (CORRECT)`

Cause: The disk block count for inode, *inode-number* is incorrect. When preening, `fsck` corrects the count.

Solution: To correct the disk block count of inode, *inode-number* by number of blocks in the file system, type `y` at the `CORRECT` prompt. Typing `y` at the `CORRECT` prompt replaces the block count of inode *inode-number* by number of blocks in the file system. To terminate the program, type `n`.

`LINK COUNT TABLE OVERFLOW (CONTINUE)`

Cause: There is no more room in the internal table for `fsck` containing allocated inodes with a link count of zero. If the `-o p` option is specified, the program exits and `fsck` must be completed manually.

Solution: To continue the program, type `y` at the `CONTINUE` prompt. If another allocated inode with a zero-link count is found, this error condition repeats. When this error occurs, a complete check of the file system is not possible. Run `fsck` again to recheck the file system. Increase the virtual memory available by killing some processes or increasing swap space, and then run `fsck` again. To terminate the program, type `n`.

`PARTIALLY ALLOCATED INODE I=`*inode-number* `(CLEAR)`

Cause: The inode, *inode-number* is not allocated or unallocated. If the `-o p` option is specified, the inode is cleared.

Solution: To deallocate the inode, *inode-number* by zeroing out its contents, type `y`. Typing `y` generates the `UNALLOCATED` error condition in phase 2 for each directory entry pointing to this inode. To ignore the error condition, type `n`. A no response is appropriate only if you intend to take other measures to fix the problem.

`PARTIALLY TRUNCATED INODE I=`*inode-number* `(SALVAGE)`

Cause: The `fsck` command has found inode, *inode-number* whose size is shorter than the number of fragments allocated to it. This condition occurs only if the system crashes while truncating a file. When preening the file system, `fsck` completes the truncation to the specified size.

Solution: To complete the truncation to the size specified in the inode, type `y` at the `SALVAGE` prompt. To ignore this error condition, type `n`.

`UNKNOWN FILE TYPE I=`*inode-number* `(CLEAR)`

Cause: The mode word of the inode, *inode-number* shows that the inode is not a pipe, character device, block device, regular file, symbolic link, FIFO file, or a directory inode. If the `-o p` option is specified, the inode is cleared.

Solution: To deallocate the inode, *inode-number* by zeroing its contents, results in the `UNALLOCATED` error condition in phase 2 for each directory entry pointing to this inode, type `y` at the `CLEAR` prompt. To ignore this error condition, type `n`.

Phase 1B – Rescan for More DUPS Messages

When a duplicate block is found in a file system, the following message is displayed.

block-number `DUP I=`*inode-number*

Cause: The inode, *inode-number* contains a block number, *block-number* that is already claimed by the same inode or another inode. This error condition generates the `BAD/DUP` error message in phase 2. Inodes that have overlapping blocks might be determined by examining this error condition and the `DUP` error condition in phase 1.

Solution: When a duplicate block is found, the file system is rescanned to find the inode that previously claimed that block.

Phase 2 – Check Path Names Messages

The phase 2 of the `fsck` check removes directory entries pointing to bad inodes found in phases 1 and 1B. It reports an error in the following conditions:

- Incorrect root inode mode and status
- Directory inode pointers out of range
- Directory entries pointing to bad inodes
- Directory integrity checks

When the file system is being preened using the `-o p` option, all errors in this phase terminate `fsck`, except those related to directories not being a multiple of the block size, duplicate and bad blocks, inodes out of range, and extraneous hard links.

The following messages might occur in phase 2 of `fsck` check.

`BAD INODE` *state-number* `TO DESCEND`

Cause: An `fsck` internal error has passed an invalid state, *state-number* to the routine that descends the file system directory structure. This error exits `fsck`.

Solution: If this error message is displayed, contact your local service provider.

`BAD INODE NUMBER FOR '.' I=`*inode-number* `OWNER=`*UID* `MODE=`*file-mode*
`SIZE=`*file-size* `MTIME=`*modification-time* `DIR=`*filename* `(FIX)`

Cause: A directory *inode-number* has been found whose inode number for the first entry in a directory (`.`) does not match *inode-number*.

Solution: To change the inode number for `.` and match the *inode-number*, type **`y`** at the `FIX` prompt. If you do not want to change the inode number for `.`, type **`n`**.

`BAD INODE NUMBER FOR '..' I=`*inode-number* `OWNER=`*UID* `MODE=`*file-mode*
`SIZE=`*file-size* `MTIME=`*modification-time* `DIR=`*filename* `(FIX)`

Cause: A directory *inode-number* has been found whose inode number for the second entry in a directory (`..`) does not match the parent of *inode-number*.

Solution: To set the inode number for `..` to match the parent directory of *inode-number*, type **`y`** at the `FIX` prompt. The `..` in the root inode points to itself. If you do not want to change the inode number for `..`, type **`n`**.

`BAD RETURN STATE` *state-number* `FROM DESCEND`

Cause: An `fsck` internal error has returned an impossible state, *state-number* from the routine that descends the file system directory structure. This error exits `fsck`.

Solution: If this message is displayed, contact your local service provider.

`BAD STATE` *state-number* `FOR ROOT INODE`

Cause: An internal error has assigned an impossible state, *state-number* to the `root` inode. This error exits `fsck`.

Solution: If this error message is displayed, contact your local service provider.

`BAD STATE state-number FOR INODE=`*inode-number*

Cause: An internal error has assigned an impossible state, *state-number* to inode, *inode-number*. This error exits `fsck`.

Solution: If this error message is displayed, contact your local service provider.

`DIRECTORY TOO SHORT I=`*inode-number* `OWNER=`*UID* `MODE=`*file-mode*
`SIZE=`*file-size* `MTIME=`*modification-time* `DIR=`*filename* `(FIX)`

Cause: A directory filename has been found whose size, *file-size* is less than the minimum directory size. The owner *UID*, mode *file-mode*, size *file-size*, modify time *modification-time*, directory name, and filename are displayed.

Solution: To increase the size of the directory to the minimum directory size, type `y` at the `FIX` prompt. To ignore this directory, type `n`.

`DIRECTORY filename: LENGTH` *file-size* `NOT MULTIPLE OF` *disk-block-size* `(ADJUST)`

Cause: A directory filename has been found with size, *file-size* that is not a multiple of the directory block size, *disk-block-size*.

Solution: To round up the length to the appropriate disk block size, type `y`. When preening the file system using the `-o p` option, `fsck` only displays a warning and adjusts the directory. To ignore this condition, type `n`.

`DIRECTORY CORRUPTED I=`*inode-number* `OWNER=`*UID* `MODE=`*file-mode*
`SIZE=`*file-size* `MTIME=`*modification-time* `DIR=`*filename* `(SALVAGE)`

Cause: A directory with an inconsistent internal state has been found.

Solution: To throw away all entries up to the next directory boundary (usually a 512-byte boundary), type `y` at the `SALVAGE` prompt. This drastic action can throw away up to 42 entries. You must perform this action only after other recovery efforts have failed. To skip to the next directory boundary and resume reading, but not modify the directory, type `n`.

`DUP/BAD I=`*inode-number* `OWNER=O MODE=M SIZE=`*file-size*
`MTIME=`*modification-time* `TYPE=`*filename* `(REMOVE)`

Cause: Phase 1 or phase 1B has found duplicate fragments or bad fragments associated with directory or file entry filename, inode *inode-number*. The owner *UID*, mode *file-mode*, size *file-size*, modification time *modification-time*, and directory or file name *filename* are displayed. If the `-o p` option is specified, the duplicate or bad fragments are removed.

Solution: To remove the directory or file entry filename, type `y` at the `REMOVE` prompt. To ignore this error condition, type `n`.

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

Cause: Phase 1 or phase 1B has found duplicate fragments or bad fragments in the `root` inode (usually inode-number 2 of the file system).

Solution: To clear the existing contents of the `root` inode and reallocate it, type `y` at the `REALLOCATE` prompt. The files and directories usually found in the `root` inode are recovered in phase 3 and moved to the `lost+found` directory. If the attempt to allocate the `root` fails, `fsck` exits with an error message, `CANNOT ALLOCATE ROOT INODE`. Type `n` to get the `CONTINUE` prompt. Type `y` to respond to the `CONTINUE` prompt, and ignore the `DUPS` or `BAD` error condition in the `root` inode and to continue the file system check. If the `root` inode is not correct, this might generate many other error messages. Type `n` to terminate the program.

`EXTRA '.' ENTRY I=`*inode-number* `OWNER=`*UID* `MODE=`*file-mode*
`SIZE=`*file-size* `MTIME=`*modification-time* `DIR=`*filename* `(FIX)`

Cause: More than one entry for `.` in a *inode-number* directory has been found.

Solution: To remove the extra entry for `.`, type `y` at the `FIX` prompt. To leave the directory unchanged, type `n`.

`EXTRA '..' ENTRY I=`*inode-number* `OWNER=`*UID* `MODE=`*file-mode*
`SIZE=`*file-size* `MTIME=`*modification-time* `DIR=`*filename* `(FIX)`

Cause: A directory *inode-number* has been found that has more than one entry for `..` in the parent directory.

Solution: To remove the extra entry for `..` in the parent directory, type `y` at the `FIX` prompt. To leave the directory unchanged, type `n`.

hard-link-number `IS AN EXTRANEOUS HARD LINK TO A DIRECTORY` *file-name* `(REMOVE)`

Cause: The `fsck` command has found an extraneous hard link, *hard-link-number* to a directory *file-name*. When preening using the `-o p` option, `fsck` ignores the extraneous hard links.

Solution: To delete the extraneous entry *hard-link-number*, type **`y`** at the `REMOVE` prompt. To ignore the error condition, type **`n`**.

inode-number `OUT OF RANGE I=`*inode-number* `NAME=`*filename* `(REMOVE)`

Cause: A directory entry filename has an inode number, *inode-number* that is greater than the end of the inode list. If the `-p` option of preen is specified, the inode is removed automatically.

Solution: To delete the directory entry filename, type **`y`** at the `REMOVE` prompt. To ignore the error condition, type **`n`**.

```
MISSING '.' I=inode-number OWNER=UID MODE=file-mode SIZE=file-size
MTIME=modification-time DIR=filename (FIX)
```

Cause: A directory, *inode-number* has been found whose first entry (`.`) is unallocated.

Solution: To build an entry for `.` with inode number equal to *inode-number*, type **`y`** at the `FIX` prompt. To leave the directory unchanged, type **`n`**.

```
MISSING '.' I=inode-number OWNER=UID MODE=file-mode SIZE=file-size
MTIME=modification-time DIR=filename CANNOT FIX, FIRST ENTRY IN
DIRECTORY CONTAINS filename
```

Cause: A directory, *inode-number* has been found whose first entry is filename. The `fsck` command cannot resolve this problem.

Solution: If this error message is displayed, contact your local service provider.

```
MISSING '.' I=inode-number OWNER=UID MODE=file-mode SIZE=file-size
MTIME=modification-time DIR=filename CANNOT FIX, INSUFFICIENT
SPACE TO ADD '.'
```

Cause: A directory *inode-number* has been found whose first entry is not pointing to the same directory (`.`). The `fsck` command cannot resolve the problem.

Solution: If this error message is displayed, contact your local service provider.

```
MISSING '..' I=inode-number OWNER=UID MODE=file-mode SIZE=file-size
MTIME=modification-time DIR=filename (FIX)
```

Cause: A directory, *inode-number* has been found whose second entry (`..`) is unallocated.

Solution: To build an entry for `..` with inode number equal to the parent of *inode-number*, type **`y`** at the `FIX` prompt. The `..` in the `root` inode points to itself. To leave the directory unchanged, type **`n`**.

```
MISSING '..' I=inode-number OWNER=UID MODE=file-mode SIZE=file-size
MTIME=modification-time DIR=filename CANNOT FIX, SECOND ENTRY IN
DIRECTORY CONTAINS filename
```

Cause: A directory, *inode-number* has been found whose second entry is *filename*. The `fsck` command cannot resolve this problem.

Solution: If this error message is displayed, contact your local service provider.

`MISSING '..' I=`*inode-number* `OWNER=`*UID* `MODE=`*file-mode* `SIZE=`*file-size*
`MTIME=`*modification-time* `DIR=`*filename* `CANNOT FIX, INSUFFICIENT SPACE`
`TO ADD '..'`

Cause: A directory, *inode-number* has been found whose second entry is not `..` in the parent directory. The `fsck` command cannot resolve this problem.

Solution: If this error message is displayed, contact your local service provider.

`NAME TOO LONG` *filename*

Cause: An excessively long path name has been found, which usually indicates loops in the file system name space. This error can occur if a privileged user has made circular links to directories.

Solution: Remove the circular links using the `rm` command.

`ROOT INODE UNALLOCATED (ALLOCATE)`

Cause: The `root` inode (mostly inode number 2) has no allocate-mode bits.

Solution: To allocate inode 2 as the root inode, type `y` at the `ALLOCATE` prompt. The files and directories usually found in the root inode are recovered in phase 3 and moved to the `lost+found` directory. If the attempt to allocate the root inode fails, `fsck` displays `CANNOT ALLOCATE ROOT INODE` message and exits. To terminate the program, type `n`.

`ROOT INODE NOT DIRECTORY (REALLOCATE)`

Cause: The `root` inode (mostly inode number 2) of the file system is not a directory inode.

Solution: To clear the existing contents of the root inode and reallocate it, type `y` at the `REALLOCATE` prompt. The files and directories usually found in the root inode are recovered in phase 3 and moved to the `lost+found` directory. If the attempt to allocate the root inode fails, `fsck` displays the message, `CANNOT ALLOCATE ROOT INODE` and exits. To have `fsck` prompt with `FIX` to try to repair the inode, type `n`.

`UNALLOCATED I=`*inode-number* `OWNER=`*UID* `MODE=`*file-mode* `SIZE=`*file-size*
`MTIME=`*modification-time* `type=`*filename*`(REMOVE)`

Cause: A directory or file entry filename points to an unallocated inode, *inode-number*. The owner *UID*, mode *file-mode*, size *file-size*, modify time *modification-time*, and file name *filename* are displayed.

Solution: To delete the directory entry filename, type `y` at the `REMOVE` prompt. To ignore the error condition, type `n`.

`ZERO LENGTH DIRECTORY I=`*inode-number* `OWNER=`*UID* `MODE=`*file-mode*
`SIZE=`*file-size* `MTIME=`*modification-time* `DIR=`*filename* `(REMOVE)`

Cause: A directory entry filename has a size, *file-size* that is zero. The owner *UID*, mode *file-mode*, size *file-size*, modify time *modification-time*, and directory name *filename* are displayed.

Solution: To remove the directory entry *filename*, type `y` at the `REMOVE` prompt. This results in the `BAD` or `DUP` error message in phase 4. To ignore the error condition, type `n`.

Phase 3 – Check Connectivity Messages

This phase checks the directories examined in phase 2 and reports error conditions resulting from unreferenced directories, and missing or full `lost+found` directory.

The following messages might occur in phase 3 `fsck` check.

`BAD INODE` *state-number* `TO DESCEND`

Cause: An internal error has caused an impossible state, *state-number* to be passed to the routine that descends the file system directory structure. The error terminates `fsck`.

Solution: If this occurs, contact your local service provider.

`DIR I=`*inode-number1* `CONNECTED. PARENT WAS I=`*inode-number2*

Cause: This is an advisory message indicating a directory inode, *inode-number1* was successfully connected to the `lost+found` directory. The parent inode, *inode-number2* of the directory inode, *inode-number1* is replaced by the inode number of the `lost+found` directory.

`DIRECTORY` *filename* `LENGTH` *file-size* `NOT MULTIPLE OF` *disk-block-size* `(ADJUST)`

Cause: A directory filename has been found with size, *file-size* that is not a multiple of the *disk-block-size*. This condition can recur in phase 3 if it is not corrected in phase 2.

Solution: To round up the length to the appropriate disk block size, type `y` at the `ADJUST` prompt. When preening, `fsck` displays a warning and adjusts the directory. To ignore this error condition, type `n`.

```
lost+found IS NOT A DIRECTORY (REALLOCATE)
```

Cause: The entry for `lost+found` is not a directory.

Solution: To allocate a directory inode and change the `lost+found` directory to reference it, type `y` at the `REALLOCATE` prompt. The previous inode reference by the `lost+found` directory is not cleared and it reclaims as an unreferenced inode or has its link count adjusted later in this phase. Inability to create a `lost+found` directory displays the message, `SORRY. CANNOT CREATE lost+found DIRECTORY` and aborts the attempt to link the lost inode, which generates the `UNREF` error message in phase 4. To abort the attempt to link the lost inode, which generates the `UNREF` error message in phase 4, type `n`.

```
NO lost+found DIRECTORY (CREATE)
```

Cause: There is no `lost+found` directory in the `root` directory of the file system. When preening, `fsck` tries to create a `lost+found` directory.

Solution: To create a `lost+found` directory in the `root` of the file system, type `y` at the `CREATE` prompt. This might lead to the message `NO SPACE LEFT IN / (EXPAND)`. If the `lost +found` directory cannot be created, `fsck` displays the message, `SORRY. CANNOT CREATE lost+found DIRECTORY` and aborts the attempt to link the lost inode. This in turn generates the `UNREF` error message later in phase 4. To abort the attempt to link up the lost inode, type `n` at the `CREATE` prompt..

```
NO SPACE LEFT IN /lost+found (EXPAND)
```

Cause: Another entry cannot be added to the `lost+found` directory in the `root` directory of the file system because there is no available space. When preening, `fsck` expands the `lost +found` directory.

Solution: To expand the `lost+found` directory and to make room for the new entry, type `y` at the `EXPAND` prompt. If the attempted expansion fails, `fsck` displays a message, `SORRY. NO SPACE IN lost+found DIRECTORY` and aborts the request to link a file to the `lost +found` directory. This error generates the `UNREF` error message later in phase 4. Delete any unnecessary entries in the `lost+found` directory. This error terminates `fsck` when preening is in effect. To abort the attempt to link the lost inode, type `n` at the `EXPAND` prompt.

`UNREF DIR I=`*inode-number* `OWNER=`*UID* `MODE=`*file-mode* `SIZE=`*file-size*

```
MTIME=modification-time (RECONNECT)
```

Cause: The directory inode, *inode-number* was not connected to a directory entry when the file system was traversed. The owner *UID*, mode *file-mode*, size *file-size*, and modification time *modification-time* of directory inode *inode-number* are displayed. When preening, `fsck` reconnects the non-empty directory inode if the directory size is non-zero. Otherwise, `fsck` clears the directory inode.

Solution: To reconnect the directory inode, *inode-number* into the `lost+found` directory, type `y` at the `RECONNECT` prompt. If the directory is successfully reconnected, a `CONNECTED` message is displayed. Otherwise, one of the `lost+found` error messages is displayed. To ignore this error condition, type `n` at the `RECONNECT` prompt. This error causes the `UNREF` error condition in phase 4.

Phase 4 – Check Reference Counts Messages

This phase checks the link count information obtained in phases 2 and 3. It reports error conditions resulting from:

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

All errors in this phase (except running out of space in the `lost+found` directory) are correctable when the file system is being preened.

The following messages might occur in phase 4.

```
BAD/DUP type I=inode-number OWNER=UID MODE=file-mode SIZE=file-size
MTIME=modification-time (CLEAR)
```

Cause: Phase 1 or phase 1B found duplicate blocks or bad blocks associated with file or directory inode, *inode-number*. The owner *UID*, mode *file-mode*, size *file-size*, and modification time *modification-time* of inode, *inode-number* are displayed.

Solution: To deallocate inode, *inode-number* by zeroing its contents, type `y` at the `CLEAR` prompt. To ignore this error condition, type `n`.

```
(CLEAR)
```

Cause: The inode mentioned in the `UNREF` error message preceding immediately cannot be reconnected. This message is not displayed if the file system is being preened because of lack of space to reconnect to files. This terminates `fsck`.

Solution: To deallocate the inode by zeroing out its contents, type `y` at the `CLEAR` prompt. To ignore the preceding error condition, type `n`.

`LINK COUNT type I=`*inode-number* `OWNER=`*UID* `MODE=`*file-mode*
`SIZE=`*file-size*
`MTIME=`*modification-time* `COUNT` *link-count* `SHOULD BE`
corrected-link-count `(ADJUST)`

Cause: The link count for a directory or a file inode with *inode-number* is *link-count* but must be *corrected-link-count*. The owner *UID*, mode *file-mode*, size *file-size*, and modification time *modification-time* of the inode, *inode-number* are displayed. If the `-o p` option is specified, the link count is adjusted unless the number of references is increasing. This condition does not occur unless there is a hardware failure. When the number of references is increasing during preening, `fsck` displays the `LINK COUNT INCREASING` message and exits.

Solution: To replace the link count of directory or file inode *inode-number* with *corrected-link-count*, type `y` at the `ADJUST` prompt. To ignore this error condition, type `n`.

```
lost+found IS NOT A DIRECTORY (REALLOCATE)
```

Cause: The entry for `lost+found` is not a directory.

Solution: To allocate a directory inode and change the `lost+found` directory to reference it, type `y` at the `REALLOCATE` prompt. The previous inode referenced by the `lost+found` directory is not cleared. It is either reclaimed as an unreferenced inode or has its link count adjusted later in this phase. Inability to create a `lost+found` directory displays the message, `SORRY. CANNOT CREATE lost+found DIRECTORY` and aborts the attempt to link up the lost inode. This error generates the `UNREF` error message later in phase 4. To abort the attempt to link up the lost inode, type `n` at the `REALLOCATE` prompt.

```
NO lost+found DIRECTORY (CREATE)
```

Cause: There is no `lost+found` directory in the `root` directory of the file system. When preening, `fsck` tries to create a `lost+found` directory.

Solution: To create a `lost+found` directory in the root of the file system, type `y` at the `CREATE` prompt. If the `lost+found` directory cannot be created, `fsck` displays the message, `SORRY. CANNOT CREATE lost+found DIRECTORY` and aborts the attempt to link the lost inode. This error in turn generates the `UNREF` error message later in phase 4. To abort the attempt to link up the lost inode, type `n` at the `CREATE` prompt.

`NO SPACE LEFT IN / lost+found (EXPAND)`

Cause: There is no space to add another entry to the `lost+found` directory in the `root` directory of the file system. When preening, `fsck` expands the `lost+found` directory.

Solution: To expand the `lost+found` directory in order to make room for the new entry, type `y` at the `EXPAND` prompt. If the attempted expansion fails, `fsck` displays the message, `SORRY. NO SPACE IN lost+found DIRECTORY` and aborts the request to link a file to the `lost+found` directory. This error generates the `UNREF` error message later in phase 4. Delete any unnecessary entries in the `lost+found` directory. This error terminates `fsck` when preening using `-o p` option is in effect. To abort the attempt to link up the lost inode, type `n` at the `EXPAND` prompt.

`UNREF FILE I=`*inode-number* `OWNER=`*UID* `MODE=`*file-mode* `SIZE=`*file-size*
`MTIME=`*modification-time* `(RECONNECT)`

Cause: File inode *inode-number* was not connected to a directory entry when the file system was traversed. The owner *UID*, mode *file-mode*, size *file-size*, and modification time, *modification-time* of inode, *inode-number* are displayed. When `fsck` is preening, the file is cleared if either its size or its link count is zero; otherwise, it is reconnected.

Solution: To reconnect inode, *inode-number* to the file system in the `lost+found` directory, type `y`. This error might generate the `lost+found` error message in phase 4 if there are problems connecting inode, *inode-number* to the `lost+found` directory. To ignore this error condition, type `n`. This error always invokes the `CLEAR` error condition in phase 4.

`UNREF type I=`*inode-number* `OWNER=`*UID* `MODE=`*file-mode* `SIZE=`*file-size*
`MTIME=`*modification-time* `(CLEAR)`

Cause: Inode, *inode-number* (whose type is directory or file) was not connected to a directory entry when the file system was traversed. The owner *UID*, mode *file-mode*, size *file-size*, and modification time *modification-time* of inode *inode-number* are displayed. When `fsck` is preening, the file is cleared if either its size or its link count is zero; otherwise, it is reconnected.

Solution: To deallocate inode, *inode-number* by zeroing its contents, type `y` at the `CLEAR` prompt. To ignore this error condition, type `n` at the `CLEAR` prompt.

`ZERO LENGTH DIRECTORY I=`*inode-number* `OWNER=`*UID* `MODE=`*file-mode*

`SIZE=`*file-size* `MTIME=`*modification-time*`(CLEAR)`

Cause: A directory entry filename has a size, *file-size* that is zero. The owner *UID*, mode *file-mode*, size *file-size*, modification time *modification-time*, and directory name filename are displayed.

Solution: To deallocate the directory inode, *inode-number* by zeroing out its contents, type `y`. To ignore the error condition, type `n`.

Phase 5 – Check Cylinder Groups Messages

This section contains phase 5 `fsck` messages.

This phase checks the free-fragment and used-inode maps. It reports error conditions resulting from:

- Allocated inodes missing from used-inode maps
- Free fragments missing from free-fragment maps
- Free inodes in the used-inode maps
- Incorrect total free-fragment count
- Incorrect total used inode count

The following messages might occur in phase 5.

```
FRAG BITMAP WRONG (CORRECTED)
```

Cause: A cylinder group (CG) fragment map is missing some free fragments. During preening, `fsck` reconstructs the maps.

Solution: To reconstruct the free-fragment map, type `y` at the `SALVAGE` prompt. To ignore this error condition, type `n`.

`CG` *cg-number*`: BAD MAGIC NUMBER`

Cause: The magic number of CG, *cg-number* is wrong. This error usually indicates that the CG maps have been destroyed. When running interactively, the CG is marked as needing reconstruction. `fsck` terminates if the file system is being preened.

Solution: If this occurs, contact your local service provider.

```
CORRECT GLOBAL SUMMARY (SALVAGE)
```

Cause: The summary information is incorrect. When preening, `fsck` recomputes the summary information.

Solution: To reconstruct the summary information, type `y` at the `SALVAGE` prompt. To ignore this error condition, type `n` at the `SALVAGE` prompt.

`fsck` Summary Messages

This section contains `fsck` summary messages in the current Oracle Solaris release. The following messages are displayed in the cleanup phase.

Once a file system has been checked, a few summary messages are displayed.

```
number-of files, number-of-files
used, number-of-files free (number-of frags, number-of blocks,
percent fragmentation)
```

This message indicates that the file system checked contains the number of files using the number of fragment-sized blocks, and that there are number of fragment-sized blocks free in the file system. The numbers in parentheses break the free count down into number of free fragments, number of free full-sized blocks, and the percent fragmentation.

```
***** FILE SYSTEM WAS MODIFIED *****
```

This message indicates that the file system was modified by `fsck`. There is no need to rerun `fsck` if you see this message. This message is just informational about corrective actions performed by `fsck`.

Cleanup Phase Messages

This section contains `fsck` cleanup phase messages in the Oracle Solaris 10 and later releases. In this release, similar messages can be found in the `fsck` summary phase.

Once a file system has been checked, a few cleanup functions are performed. The cleanup phase displays the following status messages.

```
number-of files, number-of-files
used, number-of-files free (number-of frags, number-of blocks,
percent fragmentation)
```

This message indicates that the file system checked contains number of files using number of fragment-sized blocks, and that there are number of fragment-sized blocks free in the file system. The numbers in parentheses break the free count down into number of free fragments, number-of free full-sized blocks, and the percent fragmentation.

```
***** FILE SYSTEM WAS MODIFIED *****
```

This message indicates that the file system was modified by `fsck`. If this file system is mounted or is the current root (/) file system, reboot. If the file system is mounted, you might need to unmount it and run `fsck` again; otherwise, the work done by `fsck` might be undone by the in-core copies of tables..

```
filename FILE SYSTEM STATE SET TO OKAY
```

This message indicates that file system, *filename* was marked as stable. Use the `fsck -m` command to determine if the file system needs checking.

```
filename FILE SYSTEM STATE NOT SET TO OKAY
```

This message indicates that file system, *filename* was not marked as stable. Use the `fsck -m` command to determine if the file system needs checking.

Fixing a UFS File System That the `fsck` Command Cannot Repair

The `fsck` command operates in several passes. For example, a problem corrected in later passes can expose other problems that have occurred in earlier passes. Therefore, it is sometimes necessary to run `fsck` until it no longer reports any problems. Doing so ensures that all errors are found and repaired.

Pay attention to the information displayed by the `fsck` command. This information might help you fix the problem. For example, the messages might point to a damaged directory. If you delete the directory, you might find that the `fsck` command runs cleanly.

If the `fsck` command still cannot repair the file system, try to use the `ff`, `clri`, and `ncheck` commands to understand and fix what is wrong. For information about these commands, see the `fsdb`(1M), `ff`(1M), `clri`(1M), and `ncheck`(1M) man pages.

If you are unable to repair a file system but can mount it as read-only, try using the `cp`, `tar`, or `cpio` commands to retrieve all or part of the data from the file system.

If hardware disk errors are causing the problem, you might need to reformat and repartition the disk again before re-creating and restoring file systems. Check if the device cables and connectors are functional before replacing the disk device. Hardware errors usually display the same error again and again across different commands. The `format` command tries to work around bad blocks on the disk. However, if the disk is too severely damaged, the problems might persist, even after you reformat the disk. For information about the `format` command, see the `format`(1M) man page.

▼ How to Restore a Bad Superblock

1. **Become an administrator.**

   For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.3*.

2. **Change to a directory outside the damaged file system and unmount the file system.**

   ```
   # unmount /mount-point
   ```

3. **Display the superblock values using the `newfs -N` command.**

   ```
   # newfs -N /dev/rdsk/device-name
   ```

   The command output displays the block numbers that were used for the superblock copies when the `newfs` command created the file system, unless the file system was created with special parameters. For information about creating a customized file system, "Customizing UFS File System Parameters" on page 17.

   Caution - Ensure to use the `-N` option. If you omit the `-N` option, you might destroy all the data in the file system and replace it with an empty file system.

4. **Provide an alternate superblock using the `fsck` command.**

   ```
   # fsck -F ufs -o b=block-number /dev/rdsk/device-name
   ```

   The `fsck` command uses an alternate superblock to restore the primary superblock. You can try 32 as an alternate block. You can use any of the alternate blocks shown by the `newfs -N` command.

Example 14 Restoring a Bad Superblock

The following example shows how to restore the superblock copy 5264.

```
# newfs -N /dev/rdsk/c0t3d0s7
/dev/rdsk/c0t3d0s7: 163944 sectors in 506 cylinders of 9 tracks, 36 sectors
 83.9MB in 32 cyl groups (16 c/g, 2.65MB/g, 1216 i/g)
```

```
super-block backups (for fsck -b #) at:
 32, 5264, 10496, 15728, 20960, 26192, 31424, 36656, 41888,
 47120, 52352, 57584, 62816, 68048, 73280, 78512, 82976, 88208,
 93440, 98672, 103904, 109136, 114368, 119600, 124832, 130064, 135296,
 140528, 145760, 150992, 156224, 161456,
# fsck -F ufs -o b=5264 /dev/rdsk/c0t3d0s7
Alternate superblock location: 5264.
** /dev/rdsk/c0t3d0s7
** Last Mounted on
** Phase 1 - Check Blocks and Sizes
** Phase 2 - Check Pathnames
** Phase 3 - Check Connectivity
** Phase 4 - Check Reference Counts
** Phase 5 - Check Cyl groups
36 files, 867 used, 75712 free (16 frags, 9462 blocks, 0.0% fragmentation)

***** FILE SYSTEM WAS MODIFIED *****
#
```

♦ ♦ ♦ CHAPTER 3

Managing Data and Free Space

When a file system runs out of free space, it leads to numerous system and application failures. When the file system gets full, you might experience the following system issues:

- Degradation in performance
- Difficulty in spotting unused blocks by the kernel to store new data
- System hangs due to long response time

The threshold when the effects on the system becomes visible varies from one file system to another. As a best practice, always keep the file system space usage below 90%.

Oracle Solaris provides the following standard UNIX tools to monitor file system space usage:

`df`	Displays the number of used blocks, number of free blocks, and calculates total file system space usage in percentage.
`du`	Displays the number of blocks a file or a set of files occupies.

File System Space Usage

The `df` tool without any options, prints the usage statistics for all mounted file systems. You can use the `-k` and `-h` options for more descriptive output. The `-k` option changes block units to 1024 bytes (effectively kilobytes) and the `-h` option generates human readable output as the amount is displayed in kilobytes, megabytes or gigabytes.

```
# df [-kh] [directory, dataset or device]
```

The `df` tool can use a directory, a ZFS dataset or a block device to filter the output. The command then prints out the statistics just for the given file system.

EXAMPLE 15 Displaying File System Space Usage

This example shows an output of the `df` tool. The output shows mount points, devices containing the file system, number of free blocks (in units of 512 bytes), and the number of files that can be stored.

```
# df
/                  (rpool/ROOT/solaris):63807702 blocks 63807702 files
/var               (rpool/ROOT/solaris/var):63807702 blocks 63807702 files
/devices           (/devices          ):       0 blocks        0 files
/dev               (/dev              ):       0 blocks        0 files
/system/contract   (ctfs              ):       0 blocks 2147483583 files
/proc              (proc              ):       0 blocks    29927 files
/etc/mnttab        (mnttab            ):       0 blocks        0 files
/system/volatile   (swap              ): 4176136 blocks  1130857 files
/tmp               (swap              ): 4176136 blocks  1130857 files
/system/object     (objfs             ):       0 blocks 2147483413 files
/etc/dfs/sharetab  (sharefs           ):       0 blocks 2147483646 files
/dev/fd            (fd                ):       0 blocks        0 files
/lib/libc.so.1     (/usr/lib/libc/libc_hwcap2.so.1):63807702 blocks 63807702 files
/var/share         (rpool/VARSHARE    ):63807702 blocks 63807702 files
/export            (rpool/export      ):63807702 blocks 63807702 files
/export/home       (rpool/export/home ):63807702 blocks 63807702 files
/export/home/jack  (rpool/export/home/jack):63807702 blocks 63807702 files
/rpool             (rpool             ):63807702 blocks 63807702 files
/system/zones      (rpool/VARSHARE/zones):63807702 blocks 63807702 files
/var/share/kvol    (rpool/VARSHARE/kvol):63807702 blocks 63807702 files
/var/share/sstore/repo(rpool/VARSHARE/sstore):63807702 blocks 63807702 files
/var/share/tmp     (rpool/VARSHARE/tmp):63807702 blocks 63807702 files
/var/share/pkg     (rpool/VARSHARE/pkg):63807702 blocks 63807702 files
/var/share/pkg/repositories(rpool/VARSHARE/pkg/repositories):63807702 blocks 63807702
 files
/mnt               (/dev/dsk/c2t2d0s0 ):107192852 blocks  8491364 files
```

EXAMPLE 16 File System Space Usage in Descriptive Format With Unit Suffixes

This example shows an output of the `df` command where numbers of blocks are translated. The output also shows the file system usage in percentages.

```
# df -h
Filesystem             Size   Used  Available Capacity  Mounted on
rpool/ROOT/solaris      67G   3.4G        30G    10%    /
rpool/ROOT/solaris/var
                        67G   574M        30G     2%    /var
/devices                 0K     0K         0K     0%    /devices
/dev                     0K     0K         0K     0%    /dev
ctfs                     0K     0K         0K     0%    /system/contract
```

```
proc                       0K      0K         0K      0%    /proc
mnttab                     0K      0K         0K      0%    /etc/mnttab
swap                     2.0G    2.5M       2.0G      1%    /system/volatile
swap                     2.0G     60K       2.0G      1%    /tmp
objfs                      0K      0K         0K      0%    /system/object
sharefs                    0K      0K         0K      0%    /etc/dfs/sharetab
fd                         0K      0K         0K      0%    /dev/fd
/usr/lib/libc/libc_hwcap2.so.1
                          34G    3.4G        30G     10%    /lib/libc.so.1
rpool/VARSHARE            67G     77M        30G      1%    /var/share
rpool/export              67G     36K        30G      1%    /export
rpool/export/home         67G     33K        30G      1%    /export/home
rpool/export/home/jack
                          67G     34K        30G      1%    /export/home/jack
rpool                     67G    3.3M        30G      1%    /rpool
rpool/VARSHARE/zones      67G     31K        30G      1%    /system/zones
rpool/VARSHARE/kvol       67G     31K        30G      1%    /var/share/kvol
rpool/VARSHARE/sstore
                          67G    279M        30G      1%    /var/share/sstore/repo
rpool/VARSHARE/tmp        67G    9.5M        30G      1%    /var/share/tmp
rpool/VARSHARE/pkg        67G     32K        30G      1%    /var/share/pkg
rpool/VARSHARE/pkg/repositories
                          67G     31K        30G      1%    /var/share/pkg/repositories
```

For more information, see the `df`(1M) man page.

Files Using the Most Space in a File System

Sorting files on a file system by their size and looking for the largest file might not be accurate. Some file systems might contain files larger than the actual size occupied on the device. Such files might seem very large though it occupies only a few blocks. To find the number of blocks allocated for each file, use the disk usage (`du`) tool.

```
# du [-sh] [path ...]
```

The disk usage tool (`du`) checks all the files in a subtree, displays the number of blocks allocated for each file, and ends the list with a total sum of blocks. If no path is specified, the `du` tool performs its operation in the current working directory.

Use the `-s` option with the `du` command to display only the total sum for the subtree. The `-h` option changes the sizes to human readable format.

EXAMPLE 17 Displaying Directory Space Usage Summary

This example shows the output of the `du` command with total number of occupied blocks of all the directories under the `/usr` subtree.

```
# du -sh /usr/*
  86K   /usr/X11
  30M   /usr/apache2
 146M   /usr/bin
   0K   /usr/dict
  44M   /usr/gnu
  49M   /usr/include
   0K   /usr/java
 157M   /usr/jdk
 1.9M   /usr/kernel
 1.3G   /usr/lib
   9K   /usr/proc
   0K   /usr/pub
 938K   /usr/sadm
  56M   /usr/sbin
  63K   /usr/sfw
 726M   /usr/share
 3.8M   /usr/xpg4
 914K   /usr/xpg6
 452K   /usr/xpg7
```

For more information, see the `du`(1) man page.

Quotas in File System

Oracle Solaris provides a mechanism to limit the amount of space a user or a group can occupy in a file system. This mechanism is called quotas. Traditionally, quotas enable you to limit the number of files or the number of blocks a user can allocate on a given file system.

Quotas are supported by UFS and ZFS file systems. NFS does not provide quotas by itself, but can propagate the information from server local file system to client.

This section focuses on UFS quotas. For information about ZFS quotas, see "Setting ZFS Quotas and Reservations" in *Managing ZFS File Systems in Oracle Solaris 11.3*.

UFS quotas limit the number of files and number of allocated blocks per user. Quotas can either be soft or hard. The soft and hard quota limits are applied to a user. When you exceed the soft quota limit, you will receive a notification but you can add more files and allocate blocks for a limited amount of time. When the time expires, you must remove the files and free the blocks until the quota limit is met. When using hard quota, the user actions are limited immediately when the limit is reached, and the response of the file system is identical to running out of space.

To activate quotas on a file system use the following commands.

```
# touch /tank/quotas
# quotaon /tank
```

Quotas are disabled automatically on mount or reboot. To disable quotas manually, use the `quotaoff` command.

```
# quotaoff mountpoint
```

The `quotaoff` command disables quotas, and enables you to create files and allocate blocks without limits.

You must enable quotas manually for each file system. By default, no limits are imposed on the file system when you enable the quotas. Once the quotas are enabled, administrators can set limits for selected users using the `edquota` command.

```
# edquota username
```

When you run the `edquota` command, it invokes an editor with line:

```
fs /tank blocks (soft = 0, hard = 0) inodes (soft = 0, hard = 0)
```

After the number is set for soft and hard quotas on the allocated blocks and files, the files are edited and saved. Once saved, the quotas are automatically applied for the specified user account.

The soft quota expiration timeouts can be set separately for each file and block using the `edquota -t` command. For example:

```
# edquota -t
```

When you run the `edquota` command, it invokes an editor with line:

```
fs /tank blocks time limit = 1 week, files time limit = 7 days
```

The timeout can be specified in months, weeks, days, hours, minutes, or seconds. By default, the timeout for hard and soft quotas are 7 days.

Quota checking can be disabled while users still have access to the file system. This might cause inconsistency in quota counting. To synchronize the quota accounting with current state of the file system, use the `quotacheck` command. For example:

```
# quotacheck mountpoint | block_device
```

As `quotacheck` reads the data directly from the device bypassing the kernel, any activity going through the kernel is not accounted. If the file system cannot be unmounted for the duration of the command, it is highly suggested to turn off logging and minimize activities on the file system.

You can also control access to the files by using file attributes and permissions. Setting permissions and file attributes allow you to protect legitimate files. You can also use BART to detect security breaches. For more information about data security and file integrity, see *Securing Files and Verifying File Integrity in Oracle Solaris 11.3*.

For more information, see the `edquota`(1M), `quotacheck`(1M), and `quotaoff`(1M) man pages.

♦ ♦ ♦ C H A P T E R 4

File System Backup and Restore

This chapter describes procedures for generic backup and file system specific backup of data.

Backing Up and Restoring UFS File Systems

This section describes how to backup and restore a UFS file system using the `ufsdump` and `ufsrestore` commands. This section also describes how to copy files and file systems using the `cpio`, `tar`, and `pax` commands.

How the `ufsdump` Command Works

The `ufsdump` command performs two passes to back up a file system. During the first pass, this command scans the raw device file for the file system, builds a table of directories and files in the memory, and writes the table to a backup media. During the second pass, the `ufsdump` command goes through the inodes in numerical order, reads the file contents and writes the data to the backup media.

The `ufsdump` command writes a sequence of fixed-size records. When the `ufsdump` command receives a notification that a record was partially written, the command assumes that it has reached the physical end of a media. This method works for most devices. If a device is not able to notify the `ufsdump` command that only a partial record has been written, a media error occurs as the `ufsdump` command tries to write another record.

The `ufsdump` command automatically detects the end-of-media for most devices. Therefore, you do not usually need to use the `-c`, `-d`, `-s`, and `-t` options to perform multi-volume backups.

You need to use the end-of-media options when the `ufsdump` command does not understand the way the device detects the end-of-media.

To ensure compatibility with the `ufsrestore` command, the `size` option can still force the `ufsdump` command to go to the next tape or diskette before reaching the end of the current tape or diskette.

Copying Data Using the `ufsdump` Command

The `ufsdump` command copies data only from the raw disk slice. If the file system is still active, any data in memory buffers might not be copied. The backup done by the `ufsdump` command neither copies for free blocks, nor makes an image of the disk slice. If symbolic links point to files on other slices, the link itself is copied. For more information, see the `ufsdump`(1M) man page.

Using the `-u` Option

You can use the `-u` option with the `ufsdump` command to maintain and update the `/etc/dumpdates` file. Each line in the `/etc/dumpdates` file shows the following information:

- Backed up file system
- Dump level of the last backup
- Day, date, and time of the backup

A sample of the `/etc/dumpdates` file is as follows:

```
# cat /etc/dumpdates
/dev/rdsk/c0t0d0s0              0 Wed Jul 7 13:36:17 2016
/dev/rdsk/c0t0d0s7              0 Thu Jul 8 12:53:12 2016
/dev/rdsk/c0t0d0s7              9 Thu Jul 8 13:41:48 2016
```

When performing an incremental backup, the `ufsdump` command checks the `/etc/dumpdates` file to find the date of the most recent backup of the next lower dump level. The command then copies all the files that were modified since the date of that lower-level backup to the media. After the backup is complete, a new information line which describes the backup you just completed, replaces the information line for the previous backup at that level.

Use the `/etc/dumpdates` file to verify that backups are performed. This verification is particularly important if you are having equipment problems. If a backup cannot be completed because of an equipment failure, the backup is not recorded in the `/etc/dumpdates` file.

If you want to restore an entire disk, check the `/etc/dumpdates` file for a list of the most recent dates and levels of backups so that you can determine in which tape you need to restore the entire file system.

Caution - The `/etc/dumpdates` file is a text file that can be edited. However, edit it only at your own risk. If you make changes to the file that do not match your archive tapes, you might be unable to find the tapes (or files) you need.

Using the `-f` Option

The `dump_file` argument (to the `-f` option) specifies the destination of the backup. The destination can be one of the following devices:

- Local tape drive
- Local diskette drive
- Remote tape drive
- Remote diskette drive
- Standard output

Use this argument when the destination is not the default local tape drive, `/dev/rmt/0`. If you use the `-f` option, then you must specify a value for the `dump_file` argument.

Note - The `dump_file` argument can also point to a file on a local disk or on a remote disk. If done by mistake, this usage can fill up a file system.

Local Tape or Diskette Drive

Typically, the `dump_file` argument specifies a raw device file for a tape device or a diskette. When the `ufsdump` command writes to an output device, it creates a single backup file that might span multiple tapes or diskettes.

You can specify a tape device or a diskette on your system by using a device abbreviation. The first device is always 0. For example, if you have a LTO7 tape connected to a SAS HBA, use the device name, `/dev/rmt/0`.

When you specify a tape device name, you can also type the letter `n` at the end of the name to indicate that the tape drive should not rewind after the backup is complete. For example, `/dev/rmt/0n`.

Use the `-no-rewind` option if you want to put more than one file onto the tape. If you run out of space during a backup, the tape does not rewind before the `ufsdump` command asks for a new tape.

Remote Tape or Diskette Drive

You can specify a remote tape device or a remote diskette by using the syntax *host*:*device*. The `ufsdump` command writes to the remote device when superuser on the local system has access to the remote system. If you usually run the `ufsdump` command as a superuser, the name of the local system must be included in the `/.rhosts` file on the remote system. If you specify the device as *user@host*:*device*, the `ufsdump` command tries to access the device on the remote system as the specified user. In this case, the specified user must be included in the `/.rhosts` file on the remote system.

Using Standard Output With the `ufsdump` Command

When you specify a dash (-) as the `dump_file` argument, the `ufsdump` command writes to standard output.

Note - The `-v` option (verify) cannot be used to verify a dump to standard output.

In a pipeline, you can use the `ufsdump` command to copy a file system by writing to a standard output, and use the `ufsrestore` command to read from a standard input. For example:

```
# ufsdump 0f - /dev/rdsk/c0t0d0s7 | (cd /home; ufsrestore xf -)
```

Specifying Files to Back Up

You must always include file names as the last argument on the command line. This argument specifies the source or content of the backup.

For a file system, specify the raw device file as `/dev/rdsk/c0t0d0s7`.

You can specify the file system by its mount point directory (for example, `/export/home`), as long as an entry for it exists in the `/etc/vfstab` file.

Note - When you use the `ufsdump` command to back up one or more directories or files (rather than a complete file system), a level 0 backup is done. Incremental backups do not apply.

Specifying Tape Characteristics

If you do not specify any tape characteristics, the `ufsdump` command uses a set of defaults. You can specify the tape cartridge (c), density (d), size (s), and number of tracks (t). Note that you can specify the options in any order, as long as the arguments that follow match the order of the options.

Limitations of the `ufsdump` Command

The `ufsdump` command has the following limitations:

- Automatically calculates the number of tapes or diskettes that are needed for backing up file systems. You can use the dry run mode (`-S` option) to determine how much space is needed before actually backing up file systems.
- Provides built-in error checking to minimize problems when it backs up an active file system.
- Backs up files that are remotely mounted from a server. Files on the server must be backed up on the server itself. Users are denied permission to run the `ufsdump` command on files that they own, which are located on a server.

Specifying the `ufsdump` Command Options and Arguments

This section describes the options and arguments for the `ufsdump` command. The syntax for the `ufsdump` command is as follows:

`/usr/sbin/ufsdump` *options arguments filenames*

options	Single string of one-letter option names.
arguments	Identifies option arguments and might consist of multiple strings. The option letters and their associated arguments must be in the same order.
filenames	Identifies the files to back up. These arguments must always come last, each separated by a space.

Default `ufsdump` Options

You can run the `ufsdump` without any options by using the following syntax:

```
# ufsdump filenames
```

The `ufsdump` command uses the following options and arguments, by default:

```
# ufsdump 9uf /dev/rmt/0 filenames
```

The option, `9`in the `ufsdump` command will do a level 9 incremental backup of all the modified files to the default tape drive since the previous backup.

The `-u` option is used to update the `/etc/dumpdates` file with information about the current backup. The `/etc/dumpdates` contains information about the previous backup and can decide which are the files to be backed-up.

The `f /dev/rmt/0` option sends backup to the first tape device directly connected to a Solaris system.

For more information, see the `ufsdump`(1M) man page.

The `ufsdump` Command and Security Issues

Consider the following aspects to ensure that there are no security issues with the file system:

- Require superuser access to use the `ufsdump` command.
- If you are performing centralized backups, ensure that superuser access entries are removed from the `/.rhosts` files on the clients and servers.

Specifying `ufsrestore` Options and Arguments

The syntax of the `ufsrestore` command is as follows:

```
# /usr/sbin/ufsrestore options arguments filenames
```

options	Single string of one-letter option names. You must include only one option.
arguments	Follows the option string with the arguments that match the options. The option letters and their associated arguments must be in the same order.
filenames	Specifies the file or files to be restored as arguments to the `-x` or `-t` options. These arguments must always come last, separated by spaces.

For more information, see the `ufsrestore`(1M) man page.

Copying Files and File System

This section includes information about the commands that you can use to copy files and file systems. It also contains information about how to copy files using different commands.

Commands for Copying File Systems

If you want to copy or move individual files, portions of file systems, or complete file systems, follow the procedures described in this section.

The following table describes the various backup and restore commands that are available in Oracle Solaris. For enterprise environments, consider using an enterprise-level backup product. Information about enterprise-level backup products is available from Oracle Technical Resources.

The following table describes the advantages and disadvantages of some of these commands.

TABLE 1 Advantages and Disadvantages of tar, pax, and cpio Commands

Command	Function	Advantages	Disadvantages
`tar`	Copies files and directory subtrees to a single tape.	■ Available on most UNIX operating systems ■ Public domain versions are readily available	■ Is not aware of file system boundaries ■ Length of full path name cannot exceed 255 characters ■ Cannot be used to create multiple tape volumes
`pax`	Copies files, special files, or file systems that require multiple tape volumes. Or, copies files to and from POSIX-compliant systems.	■ Better portability than the `tar` or `cpio` commands for POSIX-compliant systems ■ Multiple vendor support	Same disadvantages as the `tar` command, except that the `pax` command can create multiple tape volumes.
`cpio`	Copies files, special files, or file systems that require multiple tape volumes. Or, copies files from systems running latest Oracle Solaris version to systems running older Solaris version.	■ Packs data onto tape more efficiently than the `tar` command ■ Skips over any bad spots in a tape when restoring ■ Provides options for writing files with different header formats, such as `tar`, `ustar`, `crc`, `odc`, `bar`, for portability between different system types	The command syntax is more difficult than the `tar` or `pax` commands.

Command	Function	Advantages	Disadvantages
		■ Creates multiple tape volumes	

The following sections describes step-by-step instructions and examples of how to use these commands.

Copying Directories Between File Systems Using `cpio` Command

You can use the `cpio` (copy in and out) command to copy individual files, groups of files, or complete file systems. This section describes how to use the `cpio` command to copy complete file systems.

The `cpio` command is an archiving program that copies a list of files into a single, large output file. This command inserts headers between the individual files to facilitate recovery. You can use the `cpio` command to copy complete file systems to another slice, another system, or to a media device, such as a tape or diskette.

Because the `cpio` command recognizes end-of-media and prompts you to insert another volume, it is the most effective command, other than `ufsdump`, to create archives that require multiple tapes or diskettes.

With the `cpio` command, you can use the `ls` and `find` commands to list and select the files you want to copy, and then to pipe the output to the `cpio` command.

▼ How to Copy Directories Between File Systems Using the `cpio` Command

1. **Change to the appropriate directory.**

```
# cd filesystem1
```

2. **Copy the directory tree from *filesystem1* to *filesystem2* by using a combination of the `find` and `cpio` commands.**

```
# find . -print -depth | cpio -pdm filesystem2
```

`-print`	Prints the file names

`-depth`	Descends the directory hierarchy and prints file names from the bottom up
`-p`	Creates a list of files
`-d`	Creates directories as needed
`-m`	Sets the correct modification times on directories

The files from the directory name that is specified are copied. The symbolic links are preserved.

You can also specify the `-u` option to perform unconditional copy. Otherwise, older files do not replace newer files. This option might be useful if you want an exact copy of a directory, and some of the files being copied might already exist in the target directory.

For more information, see the `cpio(1)` man page.

3. **Verify that the copy was successful by displaying the contents of the destination directory.**

```
# cd filesystem2
# ls
```

4. **If required, remove the source directory.**

```
# rm -rf filesystem1
```

Example 18 Copying Directories Between File Systems Using the `cpio` Command

This example shows how to copy directories between file systems using the `cpio` command.

```
# cd /data1
# find . -print -depth | cpio -pdm /data2
19013 blocks
# cd /data2
# ls
# rm -rf /data1
```

Copying Files and File Systems to a Tape Device

You can use the `tar`, `pax`, and `cpio` commands to copy files and file systems to a tape device. You can choose the command based on the amount of flexibility and precision you require for the copy. Because all three commands use the raw device, you do not need to format or create a file system on tapes before you use them. The tape drive and device name that you use depend on the hardware configuration for each system. For more information about tape devices, see "Choosing Which Media to Use" in *Managing Devices in Oracle Solaris 11.3*.

Copying Files to a Tape Device Using the `tar` Command

Ensure that you know the following aspects before you copy files to tape using the `tar` command:

- Copying files to a tape with the `-c` option to the `tar` command destroys any files already on the tape, at or beyond the current tape position.
- You can use file name substitution wildcards (`?` and `*`) as a part of the file name that you specify when copying files. For example, to copy all documents with `.doc` extension, type `*.doc` as the file name argument.
- You cannot use file name substitution wildcards when you extract files from a `tar` archive.

For more information, see the `tar`(1) man page.

▼ How to Copy Files to a Tape Device Using the `tar` Command

1. **Change to the directory that contains the files you want to copy.**

2. **Insert a write-enabled tape into the tape drive.**

3. **Copy the files to the tape device.**

```
$ tar cvf /dev/rmt/n filenames
```

`c`	Creates an archive.
`v`	Displays the name of each file as it is archived.
`f /dev/rmt/n`	Indicates that the archive should be written to the specified device or file.
filenames	Indicates the files and directories that you want to copy. You can copy multiple files by specifying the file names with spaces. The file names that you specify are copied to the tape, overwriting any existing files on the tape.

4. **Remove the tape from the drive. Write the names of the files on the tape label.**

5. **Verify that the files are copied to the tape.**

```
$ tar tvf /dev/rmt/n
```

Example 19 Copying Files to a Tape Device Using the `tar` Command

The following example shows how to copy three files to the tape in tape drive 0 using the `tar` command.

```
$ cd /export/home/kryten
$ ls reports
reportA reportB reportC
$ tar cvf /dev/rmt/0 reports
a reports/ 0 tape blocks
a reports/reportA 59 tape blocks
a reports/reportB 61 tape blocks
a reports/reportC 63 tape blocks
$ tar tvf /dev/rmt/0
```

▼ How to List All the Files on a Tape Device Using the `tar` Command

1. **Insert a tape device into the tape drive.**

2. **Display the tape contents.**

```
$ tar tvf /dev/rmt/n
```

`-t`	Lists the table of contents for the files on the tape.
`-v`	Provides detailed information about the files on the tape, used with the `-t` option.
`-f /dev/rmt/n`	Indicates the tape device.

Example 20 Listing the Files on a Tape Device Using the `tar` Command

The following example shows a listing of files on the tape in drive 0.

```
$ tar tvf /dev/rmt/0
drwxr-xr-x   0/0        0 Jul 14 13:50 2010 reports/
-r--r--r--   0/0   206663 Jul 14 13:50 2010 reports/reportC
-r--r--r--   0/0   206663 Jul 14 13:50 2010 reports/reportB
-r--r--r--   0/0   206663 Jul 14 13:50 2010 reports/reportA
```

▼ How to Retrieve Files From a Tape Device Using the `tar` Command

1. **Change to the directory where you want to save the retrieved files.**

2. **Insert the tape into the tape drive.**

```
$ tar xvf /dev/rmt/n [filenames]
```

`-x`	Indicates that the files must be extracted from the specified archive file. All files on the tape in the specified drive are copied to the current directory.
`-v`	Displays the name of each file as it is retrieved.
`-f /dev/rmt/n`	Indicates the tape device that contains the archive.
filenames	Specifies a file to retrieve. You can retrieve multiple files by specifying the file names with spaces.

3. **Verify that the files are copied.**

```
$ ls -l
```

Example 21 Retrieving Files From a Tape Device Using the `tar` Command

The following example shows how to retrieve all the files from the tape in drive 0 using the `tar` command.

```
$ cd /var/tmp
$ tar xvf /dev/rmt/0
x reports/, 0 bytes, 0 tape blocks
x reports/reportA, 0 bytes, 0 tape blocks
x reports/reportB, 0 bytes, 0 tape blocks
x reports/reportC, 0 bytes, 0 tape blocks
x reports/reportD, 0 bytes, 0 tape blocks
$ ls -l
```

The names of the files extracted from the tape must exactly match the names of the files that are stored on the archive. If you have any doubts about the names or paths of the files, first list the files on the tape.

For more information, see the `tar`(1) man page.

Copying Files to a Tape Device Using the `pax` Command

This section describes how to copy files to a tape device using the `pax` command.

▼ How to Copy Files to a Tape Device Using the `pax` Command

1. **Change to the directory that contains the files you want to copy.**
2. **Insert a write-enabled tape into the tape drive.**
3. **Copy the files to the tape.**

   ```
   $ pax -w -f /dev/rmt/n filenames
   ```

`-w`	Enables the write mode.
`-f /dev/rmt/n`	Identifies the tape drive.
filenames	Indicates the files and directories that you want to copy. You can copy multiple files by specifying the file names with spaces.

4. **Verify that the files are copied to the tape device.**

   ```
   $ pax -f /dev/rmt/n
   ```

5. **Remove the tape device from the drive. Write the names of the files on the tape label.**

Example 22 Copying Files to a Tape Device Using the `pax` Command

The following example shows how to use the `pax` command to copy all the files in the current directory.

```
$ pax -w -f /dev/rmt/0 .
$ pax -f /dev/rmt/0
filea fileb filec
```

Copying Files to a Tape Device Using the `cpio` Command

This section describes how to copy files to a tape device using the `cpio` command.

▼ How to Copy All the Files in a Directory to a Tape Device Using the `cpio` Command

1. **Change to the directory that contains the files you want to copy.**
2. **Insert a write-enabled tape into the tape drive.**

3. **Copy the files to the tape device.**

```
$ ls | cpio -oc > /dev/rmt/n
```

`ls`	Provides the `cpio` command with a list of file names.
`-o`	Specifies that the `cpio` command should operate in copy-out mode. This option ensures portability to other systems of the vendors.
`-c`	Specifies that the `cpio` command must write header information in ASCII character format. This option ensures portability to other systems of the vendors.
`> /dev/rmt/n`	Specifies the output file.

All the files in the directory are copied to the tape in the drive you specify, overwriting any existing files on the tape. The total number of blocks that are copied is displayed in the output.

4. **Verify that the files are copied to the tape device.**

```
$ cpio -civt < /dev/rmt/n
```

`-c`	Specifies that the `cpio` command must read files in ASCII character format.
`-i`	Specifies that the `cpio` command must operate in copy-in mode, even though the command is only listing files at this point.
`-v`	Displays the output in a format that is similar to the output from the `ls -l` command.
`-t`	Lists the table of contents for the files on the tape in the tape drive that you specify.
`< /dev/rmt/n`	Specifies the input file of an existing `cpio` archive.

5. **Remove the tape from the drive. Write the names of the files on the tape label.**

Example 23 Copying All the Files in a Directory to a Tape a Device Using the `cpio` Command

The following example shows how to copy all the files in the `/export/home/kryten` directory to a tape device in the tape drive 0.

```
$ cd /export/home/kryten
$ ls | cpio -oc > /dev/rmt/0
1280 blocks
$ cpio -civt < /dev/rmt/0
-r--r--r--    1 kryten   staff      206663 Jul 14 13:52 2010, filea
-r--r--r--    1 kryten   staff      206663 Jul 14 13:52 2010, fileb
-r--r--r--    1 kryten   staff      206663 Jul 14 13:52 2010, filec
drwxr-xr-x    2 kryten   staff           0 Jul 14 13:52 2010, letters
drwxr-xr-x    2 kryten   staff           0 Jul 14 13:52 2010, reports
1280 blocks
```

For more information, see the `cpio(1)` man page.

▼ How to List the Files on a Tape Device Using the `cpio` Command

This procedure shows how to list the files on a tape device using the `cpio` command.

Note - Listing the table of contents on a tape takes a long time because the `cpio` command must process the entire archive.

---

1. **Insert an archive tape into the tape drive.**

2. **List the files on the tape.**

```
$ cpio -civt < /dev/rmt/n
```

Example 24 Listing the Files on a Tape Device Using the `cpio` Command

The following example shows how to list the files on the tape in drive 0.

```
$ cpio -civt < /dev/rmt/0
-r--r--r--    1 kryten   staff      206663 Jul 14 13:52 2010, filea
-r--r--r--    1 kryten   staff      206663 Jul 14 13:52 2010, fileb
-r--r--r--    1 kryten   staff      206663 Jul 14 13:52 2010, filec
drwxr-xr-x    2 kryten   staff           0 Jul 14 13:52 2010, letters
drwxr-xr-x    2 kryten   staff           0 Jul 14 13:52 2010, reports
1280 blocks
```

▼ How to Retrieve All the Files From a Tape Device Using the `cpio` Command

If the archive is created using relative path names, the input files are built as a directory within the current directory when you retrieve the files. If the archive is created with absolute path names, the same absolute paths are used to recreate the file on your system.

Caution - The use of absolute path names can be dangerous because you might overwrite existing files on your system.

1. **Change to the directory where you want to save the retrieved files.**

2. **Insert the tape into the tape drive.**

3. **Extract all the files from the tape device.**

   ```
   $ cpio -icvd < /dev/rmt/n
   ```

`-i`	Extracts files from standard input.
`-c`	Specifies that the `cpio` command must read files in ASCII character format.
`-v`	Displays the files as they are retrieved in a format that is similar to the output from the `ls` command.
`-d`	Creates directories as needed.
`< /dev/rmt/n`	Specifies the output file.

4. **Verify that the copy was successful by displaying the contents of the destination directory.**

   ```
   $ ls -l
   ```

Example 25 Retrieving All the Files From a Tape Device Using the `cpio` Command

The following example shows how to retrieve all the files from the tape in drive 0.

```
$ cd /var/tmp
$ cpio -icvd < /dev/rmt/0
answers
sc.directives
```

```
tests
8 blocks
$ ls -l
```

▼ How to Retrieve a Specific File From a Tape Device Using the `cpio` Command

This procedure shows how to retrieve a specific file from a tape device using the `cpio` command.

1. **Change to the directory where you want to save the retrieved files.**

2. **Insert the tape into the tape drive.**

3. **Retrieve a subset of files from the tape.**

   ```
   $ cpio -icv "*file" < /dev/rmt/n
   ```

`-i`	Extracts files from standard input.
`-c`	Specifies that the `cpio` command should read headers in ASCII character format.
`-v`	Displays the files as they are retrieved in a format that is similar to the output from the ls command.
*file	Specifies that all files that match the pattern are copied to the current directory. You can specify multiple patterns, but each pattern must be enclosed in double quotation marks.
`< /dev/rmt/n`	Specifies the input file.

4. **Verify that the files are copied.**

   ```
   $ ls -l
   ```

Example 26 Retrieving a Specific File From a Tape Device Using the `cpio` Command

The following example shows how to retrieve all files with the chapter suffix from the tape in drive 0.

```
$ cd /home/smith/Book
$ cpio -icv "*chapter" < /dev/rmt/0
```

```
Boot.chapter
Directory.chapter
Install.chapter
Intro.chapter
31 blocks
$ ls -l
```

For more information, see the `cpio(1)` man page.

Copying Files and File Systems to a Remote Tape Device

This section shows how to copy files and file systems to a remote tape device.

▼ How to Copy Files to a Remote Tape Device Using `tar` and `dd` Commands

1. **Configure `ssh` on the remote system so that you can access the tape drive.**

   For more information about configuring `ssh`, see *Managing Secure Shell Access in Oracle Solaris 11.3*.

2. **Change to the directory where you want to put the files.**

3. **Insert the tape into the tape drive.**

4. **Copy the files to a remote tape drive.**

```
$ tar cvf - filenames | ssh remote-host dd of=/dev/rmt/n obs=block-size
```

`-cf`	Creates a tape archive, lists the files as they are archived, and specifies the tape device.
`-v`	Provides additional information about the tar file entries.
`-` (Hyphen)	Represents a placeholder for the tape device.
filenames	Identifies the files to be copied. You can copy multiple files by specifying the file names with spaces.
`ssh | remote-host`	Pipes the output of the `tar` command to a remote system.

`dd of= /dev/rmt/n`	Represents the output device.
`obs=block-size`	Represents the blocking factor.

5. **Remove the tape from the drive. Write the names of the files on the tape label.**

Example 27 Copying Files to a Remote Tape Drive Using the `tar` and `dd` Commands

```
# tar cvf - * | ssh mercury dd of=/dev/rmt/0 obs=126b
password:
a answers/ 0 tape blocks
a answers/test129 1 tape blocks
a sc.directives/ 0 tape blocks
a sc.directives/sc.190089 1 tape blocks
a tests/ 0 tape blocks
a tests/test131 1 tape blocks
6+9 records in
0+1 records out
```

▼ How to Extract Files From a Remote Tape Device

1. **Insert the tape into the tape drive.**
2. **Change to a temporary directory.**

```
$ cd /var/tmp
```

3. **Extract the files from a remote tape device.**

```
$ ssh remote-host dd if=/dev/rmt/n | tar xvBpf -
```

`ssh remote-host`	Indicates a secure shell that is started to extract the files from the tape device by using the `dd` command.
`dd if=/dev/rmt/n`	Indicates the input device.
`| tar xvBpf -`	Pipes the output of the `dd` command to the `tar` command, which is used to restore the files.

4. **Verify that the files have been extracted.**

```
$ ls -l
```

Example 28 Extracting Files From a Remote Tape Device

```
$ cd /var/tmp
$ ssh mercury dd if=/dev/rmt/0 | tar xvBpf -
password:
x answers/, 0 bytes, 0 tape blocks
x answers/test129, 48 bytes, 1 tape blocks
20+0 records in
20+0 records out
x sc.directives/, 0 bytes, 0 tape blocks
x sc.directives/sc.190089, 77 bytes, 1 tape blocks
x tests/, 0 bytes, 0 tape blocks
x tests/test131, 84 bytes, 1 tape blocks
$ ls -l
```

For more information, see the `tar`(1) and `dd`(1M) man pages.

♦ ♦ ♦ C H A P T E R 5

Managing File System Performance

This chapter describes how to manage the file system performance using commands such as `fsstat` and `tunefs`.

Monitoring File System Performance Using `fsstat`

Starting with Oracle Solaris 11, you can use the `fsstat` command to report file system operations. You can report file system operations in multiple ways. For example, reports based on mount point or file system type.

For example, use the `fsstat` command to display all ZFS file system operations since the ZFS module is loaded:

```
$ fsstat zfs
new  name   name  attr  attr lookup rddir  read read  write write
file remov  chng   get   set    ops   ops   ops bytes   ops bytes
268K  145K 93.6K 28.0M 71.1K   186M 2.74M 12.9M 56.2G 1.61M 9.46G zfs
```

For example, use the `fsstat` command to display all file system operations since the `/export/ws` file system is mounted:

```
$ fsstat /export/ws
new  name   name  attr  attr lookup rddir  read read  write write
file remov  chng   get   set    ops   ops   ops bytes   ops bytes
0     0     0 18.1K     0  12.6M    52     0     0     0     0 /export/ws
```

The default form is to report statistical information in easy to understand values, such as GB, KB, and MB. For more information, see the fsstat(1M) man page.

Tuning UFS File System Performance Using `tunefs`

UFS provides the `tunefs` tool which enables you to modify some pf the performance related parameters of a file system. The parameters that can be modified are:

- The maximum number of logical blocks, belonging to one file, that is allocated contiguously. Contiguously allocated regions enable more data to be transferred in one I/O request. The value can be set to any positive integer number. The actual value will be the lesser of what has been specified and what the hardware supports.
- The maximum number of contiguous logical blocks any single file can allocate from a cylinder group.
- The minimum free space threshold, or the percentage of space held back from normal users.

For more information, see "Customizing UFS File System Parameters" on page 17 and the `tunefs`(1M) man page.

UFS Direct I/O Mode

File systems usually perform a lot of caching to avoid redundant I/O access to a device. This boosts performance when a block of data is needed multiple times over a short period of time. However, for some loads, the file system caching might not be effective or can even be disruptive. This usually involves applications (mostly databases) which perform caching on their own, and decide what needs to be cached in the memory for faster access. For such loads, UFS provides direct I/O mode, which disables caching on the file system level, enabling all read and write operations to be performed directly on the device. UFS then remains in control of the file system organization but data flow is managed by the application.

Direct I/O mode can be enabled for a whole file system or for each file. To enable direct I/O mode for a file system, you can use the `directio` option while mounting the file system. Direct I/O access is then used to access all the files on the mounted file system.

```
# mount -F UFS -o directio /dev/dsk/device-name/directory-name
```

Enabling direct I/O mode on separate files is useful on file systems with mixed load. To enable direct I/O access for a specific file, you must update the source code of the application when the file is opened. For more information, see the `directio`(3C) and `mount_ufs`(1M) man pages.

Index