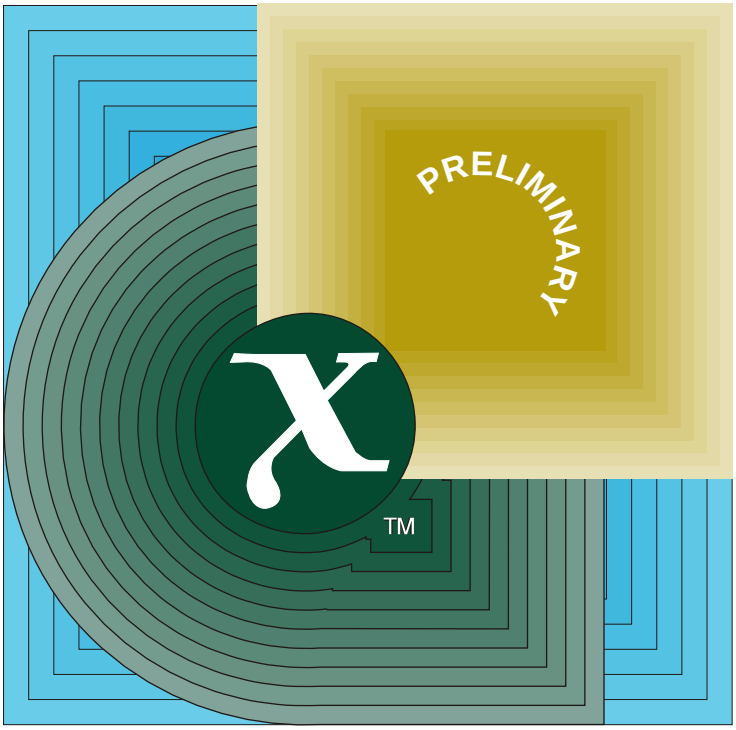
Preliminary Specification

Systems Management: File System and Scheduling Utilities (FSSU)



THE Open GROUP

[This page intentionally left blank]

X/Open Preliminary Specification

Systems Management:

File System and Scheduling Utilities (FSSU)

X/Open Company Ltd.

X/Open Preliminary Specification

Systems Management: File System and Scheduling Utilities (FSSU)

ISBN: 1-85912-145-4

X/Open Document Number: P521

Published by X/Open Company Ltd., U.K.

Any comments relating to the material contained in this document may be submitted to X/Open at:

X/Open Company Limited
Apex Plaza
Forbury Road
Reading
Berkshire, RG1 1AX
United Kingdom

or by Electronic Mail to:

XoSpecs@xopen.org

Contents

Preface

X/Open

X/Open is an independent, worldwide, open systems organisation supported by most of the world's largest information systems suppliers, user organisations and software companies. Its mission is to bring to users greater value from computing, through the practical implementation of open systems.

X/Open's strategy for achieving this goal is to combine existing and emerging standards into a comprehensive, integrated, high-value and usable open system environment, called the Common Applications Environment (CAE). This environment covers the standards, above the hardware level, that are needed to support open systems. It provides for portability and interoperability of applications, and so protects investment in existing software while enabling additions and enhancements. It also allows users to move between systems with a minimum of retraining.

X/Open defines this CAE in a set of specifications which include an evolving portfolio of application programming interfaces (APIs) which significantly enhance portability of application programs at the source code level, along with definitions of and references to protocols and protocol profiles which significantly enhance the interoperability of applications and systems.

The X/Open CAE is implemented in real products and recognised by a distinctive trade mark — the X/Open brand — that is licensed by X/Open and may be used on products which have demonstrated their conformance.

X/Open Technical Publications

X/Open publishes a wide range of technical literature, the main part of which is focussed on specification development, but which also includes Guides, Snapshots, Technical Studies, Branding/Testing documents, industry surveys, and business titles.

There are two types of X/Open specification:

- *CAE Specifications*

  CAE (Common Applications Environment) specifications are the stable specifications that form the basis for X/Open-branded products. These specifications are intended to be used widely within the industry for product development and procurement purposes.

  Anyone developing products that implement an X/Open CAE specification can enjoy the benefits of a single, widely supported standard. In addition, they can demonstrate compliance with the majority of X/Open CAE specifications once these specifications are referenced in an X/Open component or profile definition and included in the X/Open branding programme.

  CAE specifications are published as soon as they are developed, not published to coincide with the launch of a particular X/Open brand. By making its specifications available in this way, X/Open makes it possible for conformant products to be developed as soon as is practicable, so enhancing the value of the X/Open brand as a procurement aid to users.

- *Preliminary Specifications*

  These specifications, which often address an emerging area of technology and consequently are not yet supported by multiple sources of stable conformant implementations, are released in a controlled manner for the purpose of validation through implementation of products. A Preliminary specification is not a draft specification. In fact, it is as stable as X/Open can make it, and on publication has gone through the same rigorous X/Open development and review procedures as a CAE specification.

  Preliminary specifications are analogous to the *trial-use* standards issued by formal standards organisations, and product development teams are encouraged to develop products on the basis of them. However, because of the nature of the technology that a Preliminary specification is addressing, it may be untried in multiple independent implementations, and may therefore change before being published as a CAE specification. There is always the intent to progress to a corresponding CAE specification, but the ability to do so depends on consensus among X/Open members. In all cases, any resulting CAE specification is made as upwards-compatible as possible. However, complete upwards-compatibility from the Preliminary to the CAE specification cannot be guaranteed.

In addition, X/Open publishes:

- *Guides*

  These provide information that X/Open believes is useful in the evaluation, procurement, development or management of open systems, particularly those that are X/Open-compliant. X/Open Guides are advisory, not normative, and should not be referenced for purposes of specifying or claiming X/Open conformance.

- *Technical Studies*

  X/Open Technical Studies present results of analyses performed by X/Open on subjects of interest in areas relevant to X/Open's Technical Programme. They are intended to communicate the findings to the outside world and, where appropriate, stimulate discussion and actions by other bodies and the industry in general.

- *Snapshots*

  These provide a mechanism for X/Open to disseminate information on its current direction and thinking, in advance of possible development of a Specification, Guide or Technical Study. The intention is to stimulate industry debate and prototyping, and solicit feedback. A Snapshot represents the interim results of an X/Open technical activity. Although at the time of its publication, there may be an intention to progress the activity towards publication of a Specification, Guide or Technical Study, X/Open is a consensus organisation, and makes no commitment regarding future development and further publication. Similarly, a Snapshot does not represent any commitment by X/Open members to develop any specific products.

Versions and Issues of Specifications

As with all *live* documents, CAE Specifications require revision, in this case as the subject technology develops and to align with emerging associated international standards. X/Open makes a distinction between revised specifications which are fully backward compatible and those which are not:

- a new *Version* indicates that this publication includes all the same (unchanged) definitive information from the previous publication of that title, but also includes extensions or additional information. As such, it *replaces* the previous publication.

- a new *Issue* does include changes to the definitive information contained in the previous publication of that title (and may also include extensions or additional information). As such, X/Open maintains *both* the previous and new issue as current publications.

Corrigenda

Most X/Open publications deal with technology at the leading edge of open systems development. Feedback from implementation experience gained from using these publications occasionally uncovers errors or inconsistencies. Significant errors or recommended solutions to reported problems are communicated by means of Corrigenda.

The reader of this document is advised to check periodically if any Corrigenda apply to this publication. This may be done either by email to the X/Open info-server or by checking the Corrigenda list in the latest X/Open Publications Price List.

To request Corrigenda information by email, send a message to info-server@xopen.co.uk with the following in the Subject line:

```
request corrigenda; topic index
```

This will return the index of publications for which Corrigenda exist.

This Document

This document is based on the work of the Common Operating System Environment (known familiarly as *cose*) Core Facilities team. It defines some ‘‘*traditional*’’ utilities, primarily in the area of file system management.

The options used in this specification are in accordance with the ‘‘*common usage*’’ analysis, as derived by this Core Facilities team.

The following NFS feature group of utilities are specific to NFS files systems:

- *automount*
- *exportfs*
- the NFS options in *mount*.

Trade Marks

X/Open® is a registered trade mark, and the ‘‘X’’ device is a trade mark, of X/Open Company Limited.

Referenced Documents

The following documents are referenced in this specification:

XBD, Issue 4, Version 2

X/Open CAE Specification, August 1994, System Interface Definitions, Issue 4, Version 2 (ISBN: 1-85912-036-9, C434).

XCU, Issue 4, Version 2

X/Open CAE Specification, August 1994, Commands and Utilities, Issue 4, Version 2 (ISBN: 1-85912-034-2, C436).

XNFS

X/Open CAE Specification, October 1992, Protocols for X/Open Interworking: XNFS, Issue 4 (ISBN: 1-872630-66-9, C218).

XSH, Issue 4, Version 2

X/Open CAE Specification, August 1994, System Interfaces and Headers, Issue 4, Version 2 (ISBN: 1-85912-037-7, C435).

Chapter 1

Introduction

1.1 Core Facilities

This document is based on the work of the Common Operating System Environment (known familiarly as *cose*) Core Facilities team. It defines some *traditional* utilities, primarily in the area of file system management.

The options used in this specification are in accordance with the *common usage* analysis, as derived by this Core Facilities team.

1.2 NFS Feature Group

The following group of utilities are specific to NFS files systems:

- *automount*
- *exportfs*
- the NFS options in *mount*.

Chapter 2

Scheduling Management

The ‘‘scheduling management’’ utility *cron* is described, in X/Open man-page format.

NAME

cron - clock daemon

SYNOPSIS

`cron`

DESCRIPTION

The cron utility executes commands at specified dates and times. Regularly scheduled commands can be specified according to instructions found in *crontab* entries. Users can also submit their own *crontab* entries via the *crontab* command. Commands that are to be executed only once can be submitted by using the *at* or *batch* commands. Since the *cron* utility never exits, it should be executed only once. This is best done by running *cron* from the initialization process.

The *cron* utility only examines *crontab* entries and *at* and *batch* command files during process initialization and when a file changes. This reduces the overhead of checking for new or changed files at regularly scheduled intervals.

On the days of daylight savings (summer) time transition (in timezones and countries where daylight savings time applies), *cron* schedules commands differently than normal.

A non-existent time refers to an hour and minute that does not occur because of a daylight savings time transition (usually on a day during the Spring season). *DST-shift* refers to the offset that is applied to standard time to result in daylight savings time. This is normally one hour, but can be any combination of hours and minutes up to 23 hours and 59 minutes.

In the following description, an ambiguous time refers to an hour and minute that occurs twice in the same day because of a daylight savings time transition (usually on a day during the Autumn season). When a command is specified to run at an ambiguous time, the command is executed only once at the first occurrence of the ambiguous time.

When a command is specified to run a non-existent time, the command is executed after the specified time by an amount of time equal to the DST-shift. When such an adjustment would conflict with another time specified to run the command, the command is run only once rather than running the command twice at the same time.

For commands that are scheduled to run during all hours by specifying a * in the hour field of the *crontab* entry, the command is scheduled without any adjustment.

In the Spring, when there is a non-existent hour because of daylight savings time, a command that is scheduled to run multiple times during the non-existent hour will only be run once. For example, a command scheduled to run at 2:00 and 2:30 a.m. in the MST7MDT time zone will only run at 3:00 a.m. The command that was scheduled at 2:30 a.m. will not be run at all, instead of running at 3:30 a.m.

A history of all actions taken by *cron* is recorded in a system specific *cron log* database.

OPTIONS

None.

OPERANDS

None.

STDIN

Not used.

INPUT FILES

cron takes as input crontab files and at and batch command files.

ENVIRONMENT VARIABLES

The following environment variables affect the execution of cron:

LANG

Provide a default value for the internationalisation variables that are unset or null. If LANG is unset or null, the corresponding value from the implementation-specific default locale will be used. If any of the internationalisation variables contain an invalid setting, the utility will behave as if none of the variables had been set.

LC_ALL

If set to a non-empty string value, override the values of all the other internationalisation variables.

LC_CTYPE

Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single- as opposed to multi-byte characters in arguments and input files).

LC_MESSAGES

Determine the locale that should be used to affect the format and contents of diagnostic messages written to standard error.

ASYNCHRONOUS EVENTS

Default.

STDOUT

Not used.

STDERR

Not used.

OUTPUT FILES

None.

EXTENDED DESCRIPTION

None.

EXIT STATUS

cron always exits with a value of zero.

CONSEQUENCES OF ERRORS

Default.

APPLICATION USAGE

None.

EXAMPLES

The following examples assume that the time zone is MST7MDT. In this time zone the DST transition occurs one second before 2:00 a.m. and the DST-shift is 1 hour.

Consider the following *crontab* entries:

Minute	Hour	Month-Day	Month	Weekday	Command
0	01	*	*	*	Job_1
0	02	*	*	*	Job_2
0	03	*	*	*	Job_3
0	04	*	*	*	Job_4
0	*	*	*	*	Job_hourly
0	2,3,4	*	*	*	Multiple_1
0	2,4	*	*	*	Multiple_2

For the period of 01:00 a.m. to 04:00 a.m. on the days of DST transition, the results will be:

Job	Times Run in Fall	Times Run in Spring
Job_1	01:00 MDT	01:00 MST
Job_2	02:00 MDT	03:00 MDT
Job_3	03:00 MST	03:00 MDT
Job_4	04:00 MST	04:00 MDT
Job_hourly	01:00 MDT 02:00 MDT 02:00 MST 03:00 MST 04:00 MST	01:00 MST 03:00 MDT 04:00 MDT
Multiple_1	02:00 MDT 03:00 MST 04:00 MST	 03:00 MDT 04:00 MDT
Multiple_2	02:00 MDT 04:00 MST	03:00 MDT 04:00 MDT

SEE ALSO

at, *crontab*, and *sh* (see reference **XCU**).

Chapter 3

File System Management

The following ‘‘File System Management’’ utilities are described, in X/Open man-page format:

automount
exportfs
fsck
fsdb
fstyp
fuser
mkfs
mknod
mount
mountd
mvdir
ncheck
setmnt
showmount
umount

The following group of utilities are specific to NFS files systems:

- *automount*
- *exportfs*
- the NFS options in *mount*.

For ease of identification in the man-page definitions, these utilities are marked in the man-page header with the suffix **[NFS]**.

NAME

automount - automatically mount NFS file systems

SYNOPSIS

```
automount [ -mnTv ] [ -D name = value ] [ -f master-file ]
[ -M mount-directory ] [ -tl duration ] [ -tm interval ]
[ -tw interval ] [ directory map [ -mount-options ] ] ...
```

DESCRIPTION

The *automount* daemon transparently mounts NFS file systems (see reference **XNFS**) as needed. The set of directories which get mounted is specified through the means of automount maps. An automount map is a means of specifying both the name of the NFS directories to be mounted and their associated local mount points.

The *automount* utility monitors attempts to access an autommounted directory as specified by the automount map. It also monitors attempts to access the files or directories contained within that directory. When a file or directory is accessed, the *automount* utility mounts the appropriate NFS filesystem.

The *automount* daemon interacts with the kernel in a manner closely resembling an NFS server:

- The *automount* daemon uses the map to locate an appropriate NFS file server, exported file system, and mount options.
- It then mounts the file system in a temporary location, and replaces the file system entry for the directory or subdirectory with a symbolic link to the temporary location.
- If the file system is not accessed within an appropriate interval (five minutes by default), the daemon unmounts the file system and removes the symbolic link.
- If the specified directory has not already been created, the daemon creates it, and then removes it upon exiting.

Since name-to-location binding is dynamic, updates to an automount map are transparent to the user. This obviates the need to mount shared file systems prior to running applications that contain internally hard-coded references to files.

The directory name "/:" has a special meaning to the *automount* deamon. If specified, the *automount* daemon treats the map argument that follows as the name of a direct map. In a direct map, each entry associates the full pathname of a mount point with a remote file system to mount.

If the directory argument is a pathname, the map argument points to an indirect map. An indirect map, contains a list of the subdirectories contained within the indicated directory. With an indirect map, it is these subdirectories that are mounted automatically.

A map can be a file or a map entry in an implementation-defined name service. If the map is a file, the map argument must be a full pathname.

The -mount-options argument, when supplied, is a comma-separated list of options to the mount utility preceded by a hyphen. However, any conflicting mount options specified in the indicated map take precedence.

When automount receives a SIGHUP signal, it rereads the database of mounted file systems to update its internal record of currently mounted file systems.

OPTIONS

The automount daemon supports the XBD specification (see referenced documents), Section 10.2, Utility Syntax Guidelines, with the one exception of the -mount-options, which is maintained for historical compatibility.

The following options are supported:

-m Suppress initialization of directory-map pairs listed in the auto.master map entry in an implementation-defined directory name service.

-n Disable dynamic mounts. With this option, references through the automount daemon succeed only when the target filesystem has been previously mounted. This can be used to prevent NFS servers from cross-mounting each other.

-T Trace. Expand each NFS call and display it on the standard error.

-v Verbose. Log status messages to the system log database.

-D envar = value
Assign value to the indicated automount (environment) variable envar.

-f master-file
Read a local file for initialization, ahead of the auto.master map entry in an implementation-defined name service.

-M mount-directory
Mount temporary file systems in the named directory instead of in the default temporary location.

-tl duration
Specify a duration (in seconds) that a file system is to remain mounted when not in use. The default is 5 minutes.

-tm interval
Specify an interval (in seconds) between attempts to mount a filesystem. The default is 30 seconds.

-tw interval
Specify an interval (in seconds) between attempts to unmount filesystems that have exceeded their cached times. The default is 1 minute.

OPERANDS

The following operands are supported:

directory
A pathname to a local directory.

map
A file containing a map of subdirectories to be used within the directory-map pair.

STDIN

Not used.

INPUT FILES

Map Entry Format

A simple map entry (mapping) takes the form:

```
directory [-mount-options] location ...
```

where directory is the full pathname of the directory to mount, when used in a direct map, or the basename of a subdirectory in an indirect map. mount-options is a comma-separated list of mount options, and location specifies a remote filesystem from which the directory may be mounted.

In the simple case, location takes the form:

```
host:pathname
```

Multiple location fields can be specified, in which case automount sends multiple mount requests; automount mounts the file system from the first host that replies to the mount request. This request is first made to the local net or subnet. If there is no response, any connected server is allowed to respond. If location is specified in the form:

```
host:path:subdir
```

where host is the name of the host from which to mount the file system, path is the pathname of the directory to mount, and subdir, when supplied, is the name of a subdirectory to which the symbolic link is made. This can be used to prevent duplicate mounts when multiple directories in the same remote file system might be accessed. Assume a map for /users resembling:

```
mike server1:/users/server1:mike
dianna server1:/users/server1:dianna
```

Attempting to access a file in /users/mike causes automount to mount server1:/users/server1 and creates a symbolic link called /users/mike to the mike subdirectory in the temporarily-mounted filesystem. A subsequent file access request in /users/dianna results in automount simply creating a symbolic link that points to the dianna subdirectory because /users/server1 is already mounted. Given the map:

```
mike server1:/users/server1/mike
dianna server1:/users/server1/dianna
```

automount would have to mount the filesystem twice.

A mapping can be continued across input lines by escaping the new-line character with a backslash (\ ). Comments begin with a # and end at the subsequent new-line character.

Directory Pattern Matching

The & character is expanded to the value of the directory field for the entry in which it occurs. Given an entry of the form:

```
mike server1:/users/server1:&
```

the & expands to mike.

The * character, when supplied as the directory field, is recognized as the catch-all entry. Such an entry resolves to any entry not previously matched. For example, if the following entry appeared in the indirect map for /users:

```
* &:/users/&
```

this would allow automatic mounts in /users of any remote file system whose location could be specified as:

```
hostname:/users/hostname
```

Hierarchical Mappings

A hierarchical mapping takes the form:

```
directory [/[subdirectory] [-mount-options] location ...] ...
```

The initial / within the /[subdirectory] is required; the optional subdirectory is taken as a filename relative to the directory. If subdirectory is omitted in the first occurrence, the / refers to the directory itself.

Given the direct map entry:

```
/usr/local \
/ -ro,intr shasta:/usr/local ranier:/usr/local \
/bin -ro,intr ranier:/usr/local/bin shasta:/usr/local/bin \
/man -ro,intr shasta:/usr/local/man ranier:/usr/local/man
```

automount automatically mounts /usr/local, /usr/local/bin, and /usr/local/man, as needed, from either shasta or ranier, whichever host responded first.

Direct Maps

A direct map contains mappings for any number of directories. Each directory listed in the map is automatically mounted as needed. The direct map as a whole is not associated with any single directory.

Indirect Maps

An indirect map allows specifying mappings for the subdirectories to be mounted under the directory indicated on the command line. It also obscures local subdirectories for which no mapping is specified. In an indirect map, each directory field consists of the basename of a subdirectory to be mounted as needed.

An indirect map is specified on the command line.

Included Maps

The contents of another map can be included within a map with an entry of the form:

```
+mapname
```

mapname can either be a filename, or the name of a map entry in an implementation-defined name service , or one of the special maps described below.

Special Maps

Three special maps, -hosts, -passwd, and -null, are currently available: The -hosts map uses the *gethostbyname*( ) map to locate a remote host when the hostname is specified. This map specifies mounts of all exported file systems from any host.

For example, if the following *automount* command is already in effect:

```
automount /net -hosts
```

a reference to /net/hermes/usr initiates an automatic mount of all file systems from hermes that *automount* can mount, and any subsequent references to a directory under /net/hermes refer to the corresponding directory on hermes.

The -passwd map uses the password database to attempt to locate a user's home directory. For example, if the following *automount* command is already in effect:

```
automount /homes -passwd
```

then if the home directory for a user has the form /dir/server/username, and the server matches the host system on which that directory resides, *automount* mounts the user's home directory as:

```
/homes/username.
```

For this map, the tilde character (~) is recognized as a synonym for username.

The *-null* map, when indicated on the command line, cancels a previous map for the directory indicated. It can be used to cancel a map given in auto.master.

Configuration and the auto.master Map

automount normally consults the auto.master NIS configuration map for a list of initial automount maps, and sets up automatic mounts for them in addition to those given on the command line. If there are duplications, the command-line arguments take precedence. This configuration database contains arguments to the automount command rather than mappings. Unless *-f* is in effect, *automount* does not look for an auto.master file on the local host.

Maps given on the command line, or those given in a local auto.master file specified with *-f* override those in the NIS auto.master map. For example, given the command:

```
automount /homes /etc/auto.homes /- /etc/auto.direct
```

and the NIS map file auto.master containing:

```
/homes -passwd
```

automount mounts home directories using the /etc/auto.homes map instead of the special -passwd map in addition to the various directories specified in the /etc/auto.direct map.

Environment variables can be used within an automount map. For example, if $HOME appears within a map, *automount* expands it to the current value of the HOME environment variable.

To protect a reference from affixed characters, surround the variable name with curly braces. Environment variables cannot appear as the key entry in maps.

ENVIRONMENT VARIABLES

The following environment variables affect the execution of automount:

LANG
: Provide a default value for the internationalization variables that are unset or null. If LANG is unset or null, the corresponding value from the implementation-specific default locale is used. If any of the internationalization variables contain an invalid setting, the utility behaves as if none of the variables had been set.

LC_ALL
: If set to a non-empty string value, override the values of all the other internationalization variables.

LC_CTYPE
: Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single- as opposed to multi-byte characters in arguments and input files).

LC_MESSAGES
: Determine the locale that should be used to affect the format and contents of diagnostic messages written to standard error.

ASYNCHRONOUS EVENTS
Default.

STDOUT
None.

STDERR
Used only for diagnostic messages.

OUTPUT FILES
None.

EXTENDED DESCRIPTION
None.

EXIT STATUS
The following exit values are returned:

0 Successful completion.

>0 An error occurred.

CONSEQUENCES OF ERRORS
Default.

APPLICATION USAGE
Do not send the SIGKILL signal (kill -9, or kill -KILL) to the *automount* daemon. Doing so causes any processes accessing mount directories served by *automount* to hang. A system reboot may be required to recover from this state.

Do not start an *automount* daemon while another is still running. If restarting *automount*, make sure the first daemon and all of its children are not running.

If a file system mounted by *automount* is unmounted by a *umount* command, *automount* should be forced to reread the file by sending the SIGHUP signal (see *kill* in the referenced **XCU** specification).

Automatically-mounted file systems are mounted with type ignore; they do not appear in the output of either the mount or df utilities (see the referenced **XCU** specification).

EXAMPLES
None.

SEE ALSO
df, *kill*, *mount*, and *passwd* in the referenced **XCU** specification.

NAME

exportfs - export and unexport directories to NFS clients

SYNOPSIS

```
exportfs

exportfs -a [ -v ]

exportfs [ -auv ]

exportfs [ -uv ] [ file ... ]

exportfs -i [ -v ] [ -o options ] [ file ... ]
```

DESCRIPTION

The *exportfs* utility makes a local pathname (file) to a directory or file available to NFS clients (see reference **XNFS**) for mounting over the network.

The *exportfs* utility is normally invoked at boot time, and uses information contained in the exported file systems database to export file. The user with privileged authority can run the *exportfs* utility at any time to alter the list or characteristics of exported files.

If no options or arguments are specified in the command line, the *exportfs* utility writes a list of currently exported files to the standard output.

OPTIONS

The following options are supported:

-a Export all files listed in the exprted file systems database, or if -u is also specified, unexport all of the currently exported files.

-i Ignore the options in the exported file systems database. Normally, exportfs consults the exported file systems database for the options associated with the exported file.

-u Unexport the indicated files.

-v Print each pathname as it is exported or unexported.

-o options

Specify a comma-separated list of optional characteristics for the file being exported. Options can be selected from among:

async

All mounts are asynchronous.

ro Export file read-only. If not specified, the file is exported read-write.

rw=hostname[:hostname]...

Export file read-mostly. Read- mostly means exported read-only to most machines, but read-write to those specified. If not specified, the file is exported read-write to all. Implementations may impose limits on the number of hostnames which can be specified.

anon=uid

If a request comes from an unknown user, use uid as the effective user ID. Users are recognized even if they do not appear in the password database. Special treatment for users of appropriate privilege is implementation defined. Anonymous access may be disabled by setting the value of anon to another system defined value different from the default value for anonymous access.

root=hostname[:hostname]...
Give privileged access only to the privileged users from a specified hostname. The default is for no hosts to be granted privileged access. Implementations may impose limits on the number of hostnames which can be specified.

access=client[:client]...
Give mount access to each client listed. A client can either be a hostname, or a netgroup. exportfs checks for each client in the list first in the host's database, then in the netgroup database. The default value allows any machine to mount the given directory.

OPERANDS

The following operand is supported:

file file is a pathname to a directory or file in the local filesystem. file must be specified as a full pathname.

STDIN

Not used.

INPUT FILES

None.

ENVIRONMENT VARIABLES

The following environment variables affect the execution of exportfs:

LANG
Provide a default value for the internationalization variables that are unset or null. If LANG is unset or null, the corresponding value from the implementation-specific default locale is used. If any of the internationalization variables contain an invalid setting, the utility behaves as if none of the variables had been set.

LC_ALL
If set to a non-empty string value, override the values of all the other internationalization variables.

LC_CTYPE
Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single- as opposed to multi-byte characters in arguments and input files).

LC_MESSAGES
Determine the locale that should be used to affect the format and contents of diagnostic messages written to standard error.

ASYNCHRONOUS EVENTS

Default.

STDOUT

The *exportfs* utility output is in the following format:

```
"%s %s0, <filename>, <options>
```

where <filename> is the pathname of a directory or file and <options> are the options used to define the characteristics of the exported <filename>.

STDERR

Used only for diagnostic messages.

OUTPUT FILES

None.

EXTENDED DESCRIPTION

A directory that resides within the same file system and is either a parent or sub-directory of a directory that is currently exported cannot be exported. For example, /usr and /usr/local cannot both be exported if they reside in the same disk partition.

If a directory is unexported, a client removed from the access list, then exported again, the client still has access to the directory until the client unmounts the directory. Removing a client from the root or rw list takes effect immediately.

The xtab database lists currently exported directories and files. This file is maintained by exportfs. To ensure that this file is always synchronous with current system data structures, do not attempt to edit it by hand.

If an NFS mounted directory is unexported by exportfs, any access by the client to the directory causes an NFS stale file handle error. However, if exportfs is used to remove a client from the access list of an exported directory, an NFS stale file handle error does not result from any access by the client to the directory.

EXIT STATUS

The following exit values are returned:

0 Successful completion.

>0 An error occurred.

CONSEQUENCES OF ERRORS

Default.

APPLICATION USAGE

Files cannot be NFS mounted unless they are first exported by exportfs.

EXAMPLES

1. To list currently exported directories and files, enter:

   ```
   exportfs
   ```

2. To export all entries in the exported file systems database, enter:

   ```
   exportfs -a
   ```

3. To unexport all exported files and directories, enter:

   ```
   exportfs -ua
   ```

4. To unexport all exported files and directories and print each directory or filename as it is unexported, enter:

   ```
   exportfs -uav
   ```

5. To export /usr to the world, ignoring options in the exported file systems database, enter:

   ```
   exportfs -i /usr
   ```

6. To export /usr/bin and /usr/adm read-only to the world, enter:

   ```
   exportfs -i -o ro /usr/bin /usr/adm
   ```

7. To export /usr/bin read-write only to systems polk and vanness, enter:

   ```
   exportfs -i -o rw=polk:vanness /usr/bin
   ```

8. To export root access on /usr/adm only to the system named pine, and mount access to both pine and geary, enter:

```
exportfs -i -o root=pine, access=pine:geary /usr/adm
```

SEE ALSO

showmount, and *exports* and *netgroup* in the referenced **XCU** specification.

NAME

fsck - file system consistency check and interactive repair

SYNOPSIS

```
fsck [ -n ] [ -y ] [ file ... ]
```

DESCRIPTION

The *fsck* utility audits and interactively repairs inconsistent conditions for file systems on devices identified by file. If the file system is consistent, the number of files on that file system and the number of used and free 512-byte units are reported. If the file system is inconsistent, *fsck* provides a mechanism to fix these inconsistencies, depending on which form of the *fsck* utility is used.

The *fsck* utility checks a default set of file systems or the file systems specified in the command line.

The effect of the *fsck* utility on mounted file systems is implementation-defined.

OPTIONS

The following options are supported:

-y Assume a yes response to all questions asked by fsck.

-n Assume a no response to all questions asked by fsck; do not open the file system for writing.

OPERANDS

The following operand is supported:

file A special file representing the device containing the file system. If file is a pathname other than the special file representing the device containing the file system, the results are unspecified.

STDIN

Used to read an input line in response to each prompt specified in STDOUT.

INPUT FILES

None.

ENVIRONMENT VARIABLES

The following environment variables affect the execution of fsck:

LANG
: Provide a default value for the internationalization variables that are unset or null. If LANG is unset or null, the corresponding value from the implementation-specific default locale is used. If any of the internationalization variables contain an invalid setting, the utility behaves as if none of the variables had been set.

LC_ALL
: If set to a non-empty string value, override the values of all the other internationalization variables.

LC_CTYPE
: Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single- as opposed to multi-byte characters in arguments and input files).

LC_MESSAGES
: Determine the locale that should be used to affect the format and contents of diagnostic messages written to standard error.

ASYNCHRONOUS EVENTS
Default.

STDOUT
Informative text and prompts are written to standard output.

STDERR
Used only for diagnostic messages.

OUTPUT FILES
None.

EXTENDED DESCRIPTION
None.

EXIT STATUS
The following exit values are returned:

0 Successful completion.

>0 An error occurred.

CONSEQUENCES OF ERRORS
Default.

APPLICATION USAGE
The *-y* option should be used with great caution because this is a free license to continue after essentially unlimited trouble has been encountered.

Be sure the system is in single-user state before running *fsck* (see the *shutdown* utility).

EXAMPLES
None.

SEE ALSO
fsdb, *mkfs*, *mount*.

NAME

fsdb - file system debugger

SYNOPSIS

```
fsdb file
```

DESCRIPTION

The *fsdb* utility is a debugger used for file system examination or repair. Debugging is performed through use of subcommands as described in Extended Description, below. Further details of the *fsdb* utility are implementation-defined.

OPTIONS

None.

OPERANDS

The following operand is supported:

file A special file representing the device containing the file system. If file is a pathname other than the special file representing the device containing the file system, the results are unspecified.

STDIN

Used for subcommand input (see Extended Description, below).

INPUT FILES

None.

ENVIRONMENT VARIABLES

The following environment variables affect the execution of fsdb:

LANG
Provide a default value for the internationalization variables that are unset or null. If LANG is unset or null, the corresponding value from the implementation-specific default locale is used. If any of the internationalization variables contain an invalid setting, the utility behaves as if none of the variables had been set.

LC_ALL
If set to a non-empty string value, override the values of all the other internationalization variables.

LC_CTYPE
Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single- as opposed to multi-byte characters in arguments and input files).

LC_MESSAGES
Determine the locale that should be used to affect the format and contents of diagnostic messages written to standard error.

LC_TIME
Determine the format and contents of date and time strings displayed by the *fsdb* utility.

TZ Determine the timezone used with date and time strings.

ASYNCHRONOUS EVENTS

Default.

STDOUT

The output facilities generate a formatted output in various styles. Octal numbers are prefixed with a zero. Hexadecimal numbers are prefixed with 0x. It advances with the output and is left at the address of the last item written. If a number follows the p symbol, that many entries are

written. The output options available are:

i output as file attribute information

d output as directories.

Dots, tabs, and spaces can be used as function delimiters but are not necessary. A line with just a new-line character increments the current address by the size of the data type last written. That is, the address is set to the next byte, word, double word, directory entry, or inode, allowing the user to step through a region of a file system. Information is written in a format appropriate to the data type. Bytes, words, and double words are written with the address followed by the value in decimal. A .B or .D is appended to the address for byte and double-word values, respectively. Directories are written as a directory entry offset followed by the decimal file serial number and the character representation of the entry name. File attribute information is written with labeled fields describing each element.

STDERR

Used only for diagnostic messages.

OUTPUT FILES

None.

EXTENDED DESCRIPTION

The following symbols are recognized by fsdb:

#	absolute address
i	convert from file serial number to address of file attribute information
d	directory entry offset
+,–	address arithmetic
q	quit
>,<	save, restore an address
=	numerical assignment
=+	incremental assignment
=-	decremental assignment
=	character string assignment
O	error checking flip flop
p	general outout facilities
f	file output facility
B	byte mode
W	word mode
D	double-word mode
!	escape to shell.

The output options available are:

o	output as octal words
x	output as hexadecimal words

e output as decimal words

c output as characters

b output as octal bytes.

The following mnemonics are used for the examination of file attribute information and refer to the current one under examination:

at time last accessed

ct last time status changed

gid group ID number

ln link count

md mode

mt time last modified

sz file size in byte unit

uid user ID number.

The following mnemonics are used for directory examination:

di file serial number of the associated directory entry

nm name of the associated directory entry.

EXIT STATUS

The following exit values are returned:

0 Successful completion.

>0 An error occurred.

CONSEQUENCES OF ERRORS

Default.

APPLICATION USAGE

It is a recommended to execute *fsck* after having running *fsdb*.

Use of the *fsdb* utility should be limited to experienced *fsdb* users. Failure to understand fully the usage of *fsdb* and the file system's internal organization can lead to destruction of the file system and loss of data.

EXAMPLES

1. `386i` prints file serial number 386 in a file attribute information format. This now becomes the current working one.
2. `ln=4` changes the link count for the current working file attribute information to 4.
3. `ln=+1` increments the link count by 1.
4. `d7.nm="name"` changes the name field in the directory slot to the given string. Quotes are optional if the first character of the name field is alphabetic.

SEE ALSO

fsck, and the descriptions of the stat structure in **<sys/stat.h>**, and the DIR type in **<dirent.h>** in the referenced **XSH** specification.

NAME

fstyp - determine filesystem type

SYNOPSIS

```
fstyp file
```

DESCRIPTION

The *fstyp* utility allows users of appropriate privilege to determine the filesystem type of mounted or unmounted file systems using implementation-supplied heuristics programs.

If any program succeeds, its filesystem type identifier is written to standard output and the utility exits immediately. If no program succeeds, the utility writes a failure message to standard out.

OPTIONS

None.

OPERANDS

The following operand is supported:

file A special file representing the device containing the file system. If file is a pathname other than the special file representing the device containing the file system, the results are unspecified.

STDIN

Not used.

INPUT FILES

None.

ENVIRONMENT VARIABLES

The following environment variables affect the execution of ncheck:

LANG
: Provide a default value for the internationalization variables that are unset or null. If LANG is unset or null, the corresponding value from the implementation-specific default locale is used. If any of the internationalization variables contain an invalid setting, the utility behaves as if none of the variables had been set.

LC_ALL
: If set to a non-empty string value, override the values of all the other internationalization variables.

LC_COLLATE
: Determine the order of programs to be run in each of the type directories.

LC_CTYPE
: Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single- as opposed to multi-byte characters in arguments and input files).

LC_MESSAGES
: Determine the locale that should be used to affect the format and contents of diagnostic messages written to standard error.

ASYNCHRONOUS EVENTS

Default.

STDOUT

The *fstyp* utility output is in the following format:

- Successful execution:

```
<filesystem identifier>
```

- Non-successful execution:

```
Unknown_fstyp
```

where <filesystem identifier> is a filesystem type, information about which can be found in the system documentation.

STDERR

Used only for diagnostic messages.

OUTPUT FILES

None.

EXTENDED DESCRIPTION

None.

EXIT STATUS

The following exit values are returned:

0 Successful completion.

>0 An error occurred.

CONSEQUENCES OF ERRORS

Default.

APPLICATION USAGE

None.

EXAMPLES

1. To find the filesystem type for the root filesystem:

```
/etc/fstyp /dev/root
```

2. To find the filesystem type for another filesystem; here /dev/u (usually mounted on /u):

```
/etc/fstyp /dev/u
```

SEE ALSO

mount.

NAME

fuser - list process IDs of all processes that have file open

SYNOPSIS

```
fuser [ -cfku ] file ...
```

DESCRIPTION

The *fuser* utility writes to standard output the process IDs of processes that have file open. For block special devices, all processes using any file on that device are listed. In the POSIX locale, the process ID is followed by *c* if the process is using the file as its current working directory or *p* if using the file as its root directory.

OPTIONS

The following options are supported:

-c The file is treated as a mount point and the utility reports on any files open in the file system.

-f The report is only for the named file.

-u The login name, in parentheses, also follows the process ID.

-k The SIGKILL signal is sent to each process. Only users with appropriate privileges can terminate another user's process (see *kill*( ) in the referenced **XSH** specification). Lists only special files and files with set-user-ID mode.

OPERANDS

The following operand is supported:

file A pathname to the file to be special file representing the device containing the file system. If file is a pathname other than the special file representing the device containing the file system, the results are unspecified.

STDIN

Not used.

INPUT FILES

None.

ENVIRONMENT VARIABLES

The following environment variables affect the execution of fuser:

LANG

Provide a default value for the internationalization variables that are unset or null. If LANG is unset or null, the corresponding value from the implementation-specific default locale is used. If any of the internationalization variables contain an invalid setting, the utility behaves as if none of the variables had been set.

LC_ALL

If set to a non-empty string value, override the values of all the other internationalization variables.

LC_CTYPE

Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single- as opposed to multi-byte characters in arguments and input files).

LC_MESSAGES

Determine the locale that should be used to affect the format and contents of diagnostic messages written to standard error.

ASYNCHRONOUS EVENTS

Default.

STDOUT

The fuser utility output is in the following format:

```
"%d", <process_id> <process_id> ...
```

where `<process_id>` is the identifier of the process or processes that have the file open.

STDERR

Used only for diagnostic messages.

OUTPUT FILES

None.

EXTENDED DESCRIPTION

None.

EXIT STATUS

The following exit values are returned:

0 Successful completion.

>0 An error occurred.

CONSEQUENCES OF ERRORS

Default.

APPLICATION USAGE

None.

EXAMPLES

1. To terminate all processes that are preventing disk drive 1 from being unmounted, listing the process ID and login name of each process being killed, a user with appropriate privliges would enter:

   ```
   fuser -ku /dev/dsk/1s?
   ```

2. Writes to standard out the process IDs and login names of processes that have the password file open:

   ```
   fuser -u /etc/passwd
   ```

3. To combine both of the above examples into a single command line, enter:

   ```
   fuser -ku /dev/dsk/1s? -u /etc/passwd
   ```

SEE ALSO

mount, *ps* in the referenced **XCU** specification, and *kill*( ) in the referenced **XSH** specification.

NAME

mkfs - construct a file system

SYNOPSIS

```
mkfs [ -x extendedOptionString ] [ -X optionsFile ] \
     file [ extendedOperand ...]
```

DESCRIPTION

The *mkfs* utility constructs a file system by writing on file. The file system is created based on implementation-specific options and operands. The default behaviour of the *mkfs* utility is implementation-defined.

OPTIONS

The following options are supported:

-x extendedOptionString
: Used to override the value of an extended option in the defaults file.

  The extended options supported are described in ‘‘Extended Description’’, below. This option can be specified multiple times. If any extended option is defined more than once, precedence rules apply.

-X optionsFile
: Used to override the defaults specified in the system defaults file.

  The options supported are described in ‘‘Extended Description’’, below. This option can be specified multiple times. If any extended option from any file is defined more than once, precedence rules apply.

  The file has a defined ‘‘Extended Options’’ syntax".

OPERANDS

The following operands are supported:

file
: Pathname of a file on which a file system is to be constructed.

extendedOperand
: Set of implementation-defined operands.

STDIN

Not used.

INPUT FILES

Not used.

ENVIRONMENT VARIABLES

The following environment variables affect the execution of mkfs:

LANG
: Provide a default value for the internationalization variables that are unset or null. If LANG is unset or null, the corresponding value from the implementation-specific default locale is used. If any of the internationalization variables contain an invalid setting, the utility behaves as if none of the variables had been set.

LC_ALL
: If set to a non-empty string value, override the values of all the other internationalization variables.

LC_CTYPE
: Determine the locale for the interpretation of sequences of bytes of text data as characters

(for example, single- as opposed to multi-byte characters in arguments and input files).

LC_MESSAGES
Determine the locale that should be used to affect the format and contents of diagnostic messages written to standard error.

ASYNCHRONOUS EVENTS
Default.

STDOUT
Used only for informational messages.

STDERR
Used only for diagnostic messages.

OUTPUT FILES
None.

EXTENDED DESCRIPTION
The ''Extended Options'' supported are as listed below. If a default value is defined, it is listed after the option name.

allow_downdate false
Controls the ability to replace a fileset with one of a lower revision.

allow_incompatible false
Controls the ability to install software that is not compatible with the underlying operating system.

allow_multiple_versions false
Controls the ability to configure multiple versions of a product.

ask false
Controls the ability to execute **request** scripts for selected software.

autoreboot false
Controls automatic rebooting of the target host.

autorecover false
Controls automatic recovery if an error occurs during install.

autoselect_dependencies as_needed
Controls automatic dependency selection.

autoselect_dependents false
Controls automatic dependency selection.

check_contents true
Controls verification of file contents.

check_permissions true
Controls verification of file permissions.

check_requisites true
Controls verification of fileset requisites.

check_scripts true
Controls the running of the **verify** script.

check_volatile false
Controls check of volatile files.

compress_files false
Controls whether uncompressed files are to be compressed in the target distribution.

compression_type implementation_defined_value
Specifies the compression type used to compress the software files.

defer_configure false
Controls automatic configuration at install.

distribution_source_directory implementation_defined_value
Specifies the default distribution directory.

distribution_target_directory implementation_defined_value
Specifies the default distribution target.

distribution_target_serial implementation_defined_value
Specifies the default distribution target.

enforce_dependencies true
Controls the enforcement of dependency specifications.

enforce_dsa true
Controls the handling of disk space analysis errors.

enforce_locatable true
Controls the handling of errors when relocating a non-relocatable fileset.

enforce_scripts true
Controls the handling of errors generated by scripts.

files
Lists the pathnames of *file* objects to be added or deleted.

follow_symlinks false
Controls the following of symbolic links

installed_software_catalog implementation_defined_value
Specifies installed software catalog.

logfile implementation_defined_value
Specifies the location of the the logfile for the management role.

loglevel 1
Controls the amount of output sent by the utility to log files (not to stdout and stderr).

media_capacity 0
The storage capacity in megabytes of the output media.

media_type directory
The default media type.

one_liner implementation_defined_value
Specifies attributes to list.

reconfigure false
Controls reconfiguring of software.

recopy false
Controls copying of filesets.

reinstall false
Controls reinstallation of filesets.

select_local true
Controls default selection of target.

software
Specifies a default set of **software_selections** for the utility.

uncompress_files false
Controls whether compressed files are to be uncompressed in the target distribution, as specified by the value of the *compression_type* attribute of the file.

verbose 1
Controls the amount of output sent by the utility to stdout and stderr, but not to log files.

EXIT STATUS
The following exit values are returned:

EXIT STATUS
The following exit values are returned:

0 Successful completion.

>0 An error occurred.

CONSEQUENCES OF ERRORS
Default.

APPLICATION USAGE
The file operand is typically a character special device.

EXAMPLES
None.

SEE ALSO
fsck, *fsdb*, *mknod*.

NAME

mknod - create special files

SYNOPSIS

```
mknod name c major minor

mknod name b major minor

mknod name p
```

DESCRIPTION

The *mknod* utility creates the following types of files:

- character special file (first SYNOPSIS form)
- block special file (second SYNOPSIS form)
- FIFO special file, sometimes called a named pipe (third SYNOPSIS form).

Character and Block Special Files

Character special files are used for devices that can transfer single bytes at a time, such as nine-track magnetic tape drives, printers, plotters, disk drives operating in ‘‘raw’’ mode, and terminals. To create a character special file, use *c* as the second argument to the *mknod* utiltiy.

Block special files are used for devices that usually transfer a block of data at a time, such as disk drives. To create a block special file, use *b* as the second argument to *mknod*.

Only users who have appropriate privileges can use *mknod* to create a character or block special file.

FIFO Special Files

To create a FIFO (named pipe or buffer) special file, use *p* as the second argument to *mknod* (the *mkfifo* utility can also be used for this purpose - see *mkfifo* in the referenced **XCU** specification).

OPTIONS

None.

OPERANDS

The following operands are supported:

name
: The pathname of the file to be created. The newly created file has a default mode readable and writable by all users, but the mode is modified by the current setting of the user’s file mode creation mask (see umask in the XCU specification).

c To create a character special file, use *c* as the second argument to *mknod*.

b To create a block special file, use *b* as the second argument to *mknod*.

p To create a FIFO special file (named pipe), use *p* as the second argument to *mknod*.

The remaining arguments specify the device that is accessible through the new special file:

major
: The major number specifies the major device type (for example, the device driver number).

minor
: This minor number specifies the device location.

The major and minor values can each be specified in hexadecimal, octal, or decimal, using C language conventions.

STDIN

Not used.

INPUT FILES

None.

ENVIRONMENT VARIABLES

The following environment variables affect the execution of *mknod*:

LANG
: Provide a default value for the internationalization variables that are unset or null. If LANG is unset or null, the corresponding value from the implementation-specific default locale is used. If any of the internationalization variables contain an invalid setting, the utility behaves as if none of the variables had been set.

LC_ALL
: If set to a non-empty string value, override the values of all the other internationalization variables.

LC_CTYPE
: Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single- as opposed to multi-byte characters in arguments and input files).

LC_MESSAGES
: Determine the locale that should be used to affect the format and contents of diagnostic messages written to standard error.

ASYNCHRONOUS EVENTS

Default.

STDOUT

None.

STDERR

Used only for diagnostic messages.

OUTPUT FILES

None.

EXTENDED DESCRIPTION

None.

EXIT STATUS

The following exit values are returned:

0 Successful completion.

>0 An error occurred.

CONSEQUENCES OF ERRORS

Default.

APPLICATION USAGE

Assignment of major and minor numbers is specific to each system. Refer to the supplied system documentation for details.

EXAMPLES

None.

SEE ALSO

mkfifo, and *umask* in the referenced **XCU** specification.

NAME

mount - mount and file system

Note that some of the options supported by *mount* are specific to NFS (see **NFS Options** below, and also reference **XNFS**).

SYNOPSIS

```
mount [ fsname directory [ -frv ] [ -s | -u ] \
      [ -o options ] [ -t type ]]

mount -a [ -fv ] [ -s | -u ]

mount [ -p ] [ -l | -L ] [ -s | -u ]
```

DESCRIPTION

The *mount* utility directs the system to attach a removable file system **fsname** to the file tree at **directory**. **directory** must already exist, and becomes the name of the root of the newly mounted file system. **directory** must be given as an absolute path name. **fsname** must be either the name of a special file or of the form host:path. If **fsname** is of the form host:path, the file system type is assumed to be NFS (see reference **XNFS**, and the *-t* option below).

These commands maintain a table of mounted devices. If invoked with no arguments, *mount* writes the table to standard output.

OPTIONS

The following options are supported:

-a Attempt to mount all file systems described in the candidate file systems.

-f Force the file system to be mounted, even if the file system clean flag indicates that the file system should have *fsck* run on it before mounting (see *fsck* in the referenced **XCU** specification).

-p Print the list of mounted file systems in a format suitable for use in the system's list of filesystems to mount.

-r Mount the specified file system as read-only. This option implies *-o ro*. Physically write-protected file systems must be mounted in this way or errors occur when access times are updated, whether or not any explicit write is attempted.

-u Force an update of mounted file system database from the kernel mount table.

-s Do not update the mounted file system database with kernel mount information. Use of this option is provided for special cases of backward compatilibity only and is strongly discouraged. This option may be removed in a future release.

-t type
: File system type as would be determined by the fstyp utility. Acceptable types are implementation-defined. If *-a* is not used, the single file system specified is mounted as that type.

-v Verbose mode. Write a message to the standard output indicating which file system is being mounted.

-o options
: Specify a list of comma-separated options from the list below. Some options are valid for any file system type, while others apply only to a specific type.

Common Options

The following options are valid on all file systems:

defaults
Use all default options. When used, this must be the only option specified.

rw Read-write (default).

ro Read-only.

suid
Set-user-ID execution allowed (default).

nosuid
Set-user-ID execution not allowed.

NFS Options

The following options are specific to NFS file systems (see also reference **XNFS**):

bg If the first mount attempt fails, retry in the background.

fg Retry in foreground (default).

hard
Once the file system is mounted, retry subsequent NFS requests until server responds (default).

intr
Permit interrupts for hard mounts (default).

port=n
Set server UCP port number to n (default is the port customarily used for NFS servers).

retrans=n
Set number of NFS retransmissions to n (default = 4).

rsize=n
Set read buffer size to n bytes (default set by kernel).

timeo=n
Set NFS timeout to n tenths of a second (default = 7).

wsize=n
Set write buffer size to n bytes (default set by kernel).

soft
Once the file system is mounted, return error if server does not respond.

The bg option causes mount to run in the background if the server's mount daemon does not respond. *mount* attempts each request retry=n times before giving up. Once the file system is mounted, each NFS request made in the kernel waits timeo=n tenths of a second for a response. If no response arrives, the timeout is multiplied by 2 and the request is retransmitted. When retrans=n retransmissions have been sent with no reply, a soft mounted file system returns an error on the request and a hard mounted file system retries the request. By default, the retry requests for a hard mounted file system can be interrupted. If the *nointr* option is specified, retry requests for a hard mounted file system are not interruptable which means that retry requests continue until successful. File systems that are mounted rw (read-write) should use the *hard* option. The number of bytes in a read or write request can be set with the *rsize* and *wsize* options. The *devs* option allows access to devices attached to the NFS client via device files located on the

mounted NFS file system. The *nodevs* option denies access to devices attached to the NFS client by causing attempts to read or write to NFS device files to return an error.

File Attributes

The attribute cache retains file attributes on the client. Attributes for a file are assigned a time to be flushed. If the file is modified before the flush time, the flush time is extended by the time since the last modification (under the assumption that files that changed recently are likely to change soon). There is a minimum and maximum flush time extension for regular files and for directories. Setting actimeo=n extends flush time by n seconds for both regular files and directories.

OPERANDS

The following operands are supported:

fsname
: A special file representing the device containing the file system.

directory
: The mount point within the filesystem tree.

STDIN

Not used.

INPUT FILES

None.

ENVIRONMENT VARIABLES

The following environment variables affect the execution of mount:

LANG
: Provide a default value for the internationalization variables that are unset or null. If LANG is unset or null, the corresponding value from the implementation-specific default locale is used. If any of the internationalization variables contain an invalid setting, the utility behaves as if none of the variables had been set.

LC_ALL
: If set to a non-empty string value, override the values of all the other internationalization variables.

LC_CTYPE
: Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single- as opposed to multi-byte characters in arguments and input files).

LC_MESSAGES
: Determine the locale that should be used to affect the format and contents of diagnostic messages written to standard error.

ASYNCHRONOUS EVENTS

Default.

STDOUT

The *mount* utility, invoked with no options, writes information regarding each mount point, one per line. The format of the information is implementation specific, but minimally includes the mount point and the mounted-on device or directory.

STDERR

Used only for diagnostic messages.

OUTPUT FILES

None.

EXTENDED DESCRIPTION

None.

EXIT STATUS

The following exit values are returned:

0 Successful completion.

>0 An error occurred.

CONSEQUENCES OF ERRORS

Default.

APPLICATION USAGE

None.

EXAMPLES

1. To mount a local disk /dev/dsk/c0d0s4 on filesystem /usr, enter:

   ```
   mount /dev/dsk/c0d0s4 /usr
   ```

2. Mount a remote file system /usr/src from host serv on a local filesystem mount point /usr/source, enter:

   ```
   mount serv:/usr/src /usr/source -t nfs
   ```

3. Same as above:

   ```
   mount serv:/usr/src /usr/source
   ```

4. Same as above but with a soft mount; the file system is mounted read-only:

   ```
   mount serv:/usr/src /usr/source -o soft,ro
   ```

SEE ALSO

umount, *mvdir*.

NAME

mountd - mount request daemon

SYNOPSIS

```
mountd
```

DESCRIPTION

The *mountd* utility is a remote process communication daemon which serves file system mount requests. It reads the distributed file systems database to determine which directories are available to which systems. The *mountd* utility provides information on what file systems are mounted by which clients. It also updates the system table of remotely mounted filesystems for every filesystem *mount* request it serves.

The *mountd* utility is started from an implementation-specific location.

OPTIONS

None.

OPERANDS

None.

STDIN

Not used.

INPUT FILES

None.

ENVIRONMENT VARIABLES

The following environment variables affect the execution of mountd:

LANG
: Provide a default value for the internationalization variables that are unset or null. If LANG is unset or null, the corresponding value from the implementation-specific default locale is used. If any of the internationalization variables contain an invalid setting, the utility behaves as if none of the variables had been set.

LC_ALL
: If set to a non-empty string value, override the values of all the other internationalization variables.

LC_CTYPE
: Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single- as opposed to multi-byte characters in arguments and input files).

LC_MESSAGES
: Determine the locale that should be used to affect the format and contents of diagnostic messages written to standard error.

ASYNCHRONOUS EVENTS

Default.

STDOUT

None.

STDERR

Used only for diagnostic messages.

OUTPUT FILES

None.

EXTENDED DESCRIPTION

None.

EXIT STATUS

The following exit values are returned:

0 Successful completion.

>0 An error occurred.

CONSEQUENCES OF ERRORS

If a client crashes, executing *showmount* on the server may show that the client still has one or more file systems mounted.

Also, if a client mounts the same remote directory twice, only one entry appears in the table of remotely mounted filesystems. Executing *umount* on one of these directories removes the single entry and *showmount* no longer indicates that the remote directory is mounted.

APPLICATION USAGE

Information regarding the directories which are mounted by various systems can be printed using the *showmount* utility.

EXAMPLES

None.

SEE ALSO

mount, *showmount*, *exportfs*.

NAME

mvdir - move a directory

SYNOPSIS

```
mvdir directory newdirectory
```

DESCRIPTION

The *mvdir* utility moves the directory tree named by the directory operand into another existing directory (**newdirectory**) within the same file system, or renames a directory without moving it.

directory must be an existing directory.

If **newdirectory** does not exist but the directory that would contain it does, **directory** is moved and/or renamed to **newdirectory**. Otherwise, **newdirectory** must be an existing directory not already containing an entry with the same name as the last pathname component of **directory**. In this case, **directory** is moved and becomes a subdirectory of **newdirectory**. The last pathname component of **directory** is used as the name for the moved directory.

mvdir refuses to move directory if the path specified by **newdirectory** would be a descendent directory of the path specified by **directory**. Such cases are not allowed because cyclic subtrees would be created as in the case, for example, of:

```
mvdir x/y x/y/z/t
```

which is prohibited.

mvdir does not allow directory . (dot) to be moved.

Only users who have appropriate privileges can use *mvdir*.

OPTIONS

Not used.

OPERANDS

The following operands are supported:

directory
: A pathname to the directory to be moved.

newdirectory
: A pathname to the location where directory is to be moved.

STDIN

Not used.

INPUT FILES

None.

ENVIRONMENT VARIABLES

The following environment variables affect the execution of mvdir:

LANG
: Provide a default value for the internationalization variables that are unset or null. If LANG is unset or null, the corresponding value from the implementation-specific default locale is used. If any of the internationalization variables contain an invalid setting, the utility behaves as if none of the variables had been set.

LC_ALL
: If set to a non-empty string value, override the values of all the other internationalization variables.

LC_CTYPE
Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single- as opposed to multi-byte characters in arguments and input files).

LC_MESSAGES
Determine the locale that should be used to affect the format and contents of diagnostic messages written to standard error.

ASYNCHRONOUS EVENTS
Default.

STDOUT
Not used.

STDERR
Used only for diagnostic messages.

OUTPUT FILES
None.

EXTENDED DESCRIPTION
None.

EXIT STATUS
The following exit values are returned:

0 Successful completion.

>0 An error occurred.

CONSEQUENCES OF ERRORS
Default.

APPLICATION USAGE
None.

EXAMPLES
None.

SEE ALSO
cp, and *mv* in the referenced **XCU** specification.

NAME

ncheck - generate pathnames from file serial numbers

SYNOPSIS

```
ncheck [ -a ] [ -i number ] [ file ... ]

ncheck [ -s ] [ file ... ]
```

DESCRIPTION

The *ncheck* utility writes to standard output the file serial number and pathname for each specified file.

OPTIONS

The following options are supported:

-a Lists the dot (.) and dot-dot (..) pathname.

-i number
Lists only the file or files specified by the file serial number.

-s Lists only special files and files with *set-user-ID* mode.

OPERANDS

The following operand is supported:

file A special file representing the device containing the file system. If file is a pathname other than the special file representing the device containing the file system, the results are unspecified.

STDIN

Not used.

INPUT FILES

None.

ENVIRONMENT VARIABLES

The following environment variables affect the execution of ncheck:

LANG
Provide a default value for the internationalization variables that are unset or null. If LANG is unset or null, the corresponding value from the implementation-specific default locale is used. If any of the internationalization variables contain an invalid setting, the utility behaves as if none of the variables had been set.

LC_ALL
If set to a non-empty string value, override the values of all the other internationalization variables.

LC_CTYPE
Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single- as opposed to multi-byte characters in arguments and input files).

LC_MESSAGES
Determine the locale that should be used to affect the format and contents of diagnostic messages written to standard error.

ASYNCHRONOUS EVENTS

Default.

STDOUT

The *ncheck* utility output is in the following format:

```
"%d %s", <number>, <filename>
```

where <number> is the file serial number of file <filename>. Question marks (??) output as <filename> indicate a component that could not be found. Ellipses points (...) at the beginning of <filename> indicate a looprecursive condition in the <filename>.

STDERR

Used only for diagnostic messages.

OUTPUT FILES

None.

EXTENDED DESCRIPTION

None.

EXIT STATUS

The following exit values are returned:

0 Successful completion.

>0 An error occurred.

CONSEQUENCES OF ERRORS

Default.

APPLICATION USAGE

None.

EXAMPLES

1. To list the file serial number and pathname of each file in the default file systems, enter:

   ```
   ncheck
   ```

2. To list all the files in a specified file system, enter:

   ```
   ncheck -a /dev/hd1
   ```

   This lists the file serial number and pathname of each file in the /dev/hd1 file system, including the dot (.) and dot-dot (..) entries in each directory. /dev/hd1 is the name of a special file representing a file system

3. To list the name of a file when all that's known its file serial number, enter:

   ```
   ncheck -i 690 357 280 /dev/hd1
   ```

   This lists the file serial number and path name for every file in the /dev/hd1 file system with file serial numbers of 690, 357, or 280. If a file has more than one link, all of its pathnames are listed.

4. To list special and set-user-ID files, enter:

   ```
   ncheck -s /dev/hd1
   ```

   This lists thefile serial number and pathname for every file in the /dev/hd1 file system that is a special file or that has set-user-ID mode enabled.

SEE ALSO

fsck.

NAME

setmnt - establish mount table

SYNOPSIS

```
setmnt
```

DESCRIPTION

The *setmnt* utility creates the database of mounted file systems, which is needed for both the *mount* and *umount* commands (see the *mount* utility)

setmnt may silently enforce an upper limit on the maximum number of mounted file system database entries.

OPTIONS

None.

OPERANDS

None.

STDIN

The setmnt utility reads the standard input and creates an entry in the mounted file systems database for each line. Input lines have the following format:

filesys node

filesys is the name of the device special file associated with the file system (such as /dev/hd0) and **node** is the root name of that file system. Thus **filesys** and **node** become the first two strings in the mount table entry.

INPUT FILES

None.

ENVIRONMENT VARIABLES

The following environment variables affect the execution of setmnt:

LANG

Provide a default value for the internationalization variables that are unset or null. If LANG is unset or null, the corresponding value from the implementation-specific default locale is used. If any of the internationalization variables contain an invalid setting, the utility behaves as if none of the variables had been set.

LC_ALL

If set to a non-empty string value, override the values of all the other internationalization variables.

LC_CTYPE

Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single- as opposed to multi-byte characters in arguments and input files).

LC_MESSAGES

Determine the locale that should be used to affect the format and contents of diagnostic messages written to standard error.

ASYNCHRONOUS EVENTS

Default.

STDOUT

None.

STDERR

Used only for diagnostic messages.

OUTPUT FILES

None.

EXTENDED DESCRIPTION

None.

EXIT STATUS

The following exit values are returned:

0 Successful completion.

>0 An error occurred.

CONSEQUENCES OF ERRORS

Default.

APPLICATION USAGE

None.

EXAMPLES

None.

SEE ALSO

mount, *umount*.

NAME

showmount - show all remote mounts

SYNOPSIS

```
showmount [ -ade ] [ host ]
```

DESCRIPTION

The *showmount* utility lists all clients that have remotely mounted a filesystem from host. This information is maintained by the *mountd* daemon on host (see the *mountd* daemon). The default value for host is the value returned by *hostname* (see the *hostname* utility).

Warnings

If a client crashes, executing *showmount* on the server will show that the client still has a file system mounted. In other words, the client's entry is not removed from the table of remote file system mounts until the client reboots and executes:

```
umount -a
```

Also, if a client mounts the same remote directory twice, only one entry appears in /etc/rmtab. Doing a *umount* of one of these directories removes the single entry, and *showmount* no longer indicates that the remote directory is mounted.

OPTIONS

The following options are supported:

-a Write all remote mounts to standard output.

-d List the directories that have been remotely mounted by clients.

-e Write the list of exported file systems to standard output.

OPERANDS

The following operand is supported:

host
: The name of the remote host to query.

STDIN

Not used.

INPUT FILES

None.

ENVIRONMENT VARIABLES

The following environment variables affect the execution of showmount:

LANG
: Provide a default value for the internationalization variables that are unset or null. If LANG is unset or null, the corresponding value from the implementation-specific default locale is used. If any of the internationalization variables contain an invalid setting, the utility behaves as if none of the variables had been set.

LC_ALL
: If set to a non-empty string value, override the values of all the other internationalization variables.

LC_CTYPE
: Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single- as opposed to multi-byte characters in arguments and input files).

LC_MESSAGES
Determine the locale that should be used to affect the format and contents of diagnostic messages written to standard error.

ASYNCHRONOUS EVENTS
Default.

STDOUT
The *showmount* utility output is in the following format:

```
"%s %s", <hostname>, <directory>
```

where hostname is the name of the remote host, and directory is the directory or root of the file system that was mounted.

STDERR
Used only for diagnostic messages.

OUTPUT FILES
None.

EXTENDED DESCRIPTION
None.

EXIT STATUS
The following exit values are returned:

0 Successful completion.

>0 An error occurred.

CONSEQUENCES OF ERRORS
Default.

APPLICATION USAGE
None.

EXAMPLES
None.

SEE ALSO
hostname, *mount*, *mountd*, *umount*.

NAME

umount - unmount file system

SYNOPSIS

```
umount [ -v ] fsname

umount [ -v ] directory

umount -a [ -v ] [ -h host ] [ -t type ]
```

DESCRIPTION

The *umount* utility directs the system to detach the removable file system fsname previously mounted on directory directory. Either the file system name or the directory where the file system is mounted can be specified.

OPTIONS

The following options are supported:

-a Unmount all locally mounted and NFS mounted file systems described in the system table of mounted file systems.

-h host
Unmount only those file systems listed in the database of mounted file systems that are remote-mounted from host.

-t type
Unmount only file systems mounted with the given type.

-v Write a message to the standard output indicating which file system is being unmounted.

OPERANDS

The following operands are supported:

fsname
A special file representing the device containing the file system.

directory
The mount point within the filesystem tree.

STDIN

Not used.

INPUT FILES

None.

ENVIRONMENT VARIABLES

The following environment variables affect the execution of umount:

LANG
Provide a default value for the internationalization variables that are unset or null. If LANG is unset or null, the corresponding value from the implementation-specific default locale is used. If any of the internationalization variables contain an invalid setting, the utility behaves as if none of the variables had been set.

LC_ALL
If set to a non-empty string value, override the values of all the other internationalization variables.

LC_CTYPE
Determine the locale for the interpretation of sequences of bytes of text data as characters

(for example, single- as opposed to multi-byte characters in arguments and input files).

LC_MESSAGES
Determine the locale that should be used to affect the format and contents of diagnostic messages written to standard error.

ASYNCHRONOUS EVENTS
Default.

STDOUT
Not used.

STDERR
Used only for diagnostic messages.

OUTPUT FILES
None.

EXTENDED DESCRIPTION
None.

EXIT STATUS
The following exit values are returned:

0 Successful completion.

>0 An error occurred.

CONSEQUENCES OF ERRORS
Default.

APPLICATION USAGE
None.

EXAMPLES

1. To un-mount a local disk /dev/dsk/c0d0s4, enter:

```
umount /dev/dsk/c0d0s4
```

2. To un-mount a remote file system /usr/src from host server on a local filesystem mount point /usr/ source, enter:

```
umount /usr/source
```

3. Same as above:

```
mount serv:/usr/src
```

SEE ALSO
mount.