NAME

mount – Shore file systems and volumes

SYNOPSIS

```
format <device> <kb> <force>
serve <device>
unserve <device>
newvid
mkfs <device> <kb> <lvid>
rmfs <lvid>
getroot
setroot <lvid>
mount
pmount <lvid> <mountpoint>
dismount <mountpoint>
patch <mountpoint | lvid>
```

DESCRIPTION

These commands are understood by the Shore server's terminal interface, when the server is run in the foreground. The terminal interface is intended only for doing administrative functions. *It is NOT intended for normal work.* (Think of it as a machine console with blocking I/O.)

The Shore name space is a global name space and a Unix file system tree. A server can "plug" into the name space in selected places by choosing a node of the tree as its root. The server can then see everything that descends from that root.

A Shore file system is the unit of distribution of objects across devices and servers. A Shore file system is implemented as a Shore Storage Manager volume, and so a file system is identified by its volume ID.

Each file system is a hierarchy, containing a portion of the name space, and a single root node. At its leaves, a Shore file system can contain links to the roots of other Shore file systems, which can be served by the same server or by remote servers.

format, serve, unserve: Making devices usable

Use the command

```
format <device> <kb> [<force>]
```

to make the server prepare a device (a Unix file or a raw device) for use. Data that reside on the device are destroyed by this command. The second argument indicates the intended size of the device in kilobytes.

Use the command

```
serve <device>
```

to inform a server that it is the one responsible for all access to the named device. The device is identified by a Unix path. The command **unserve**reverses**the**process. A device must be served before volumes and file systems can be made on it.

mkfs, rmfs: Making file systems

To make a Shore file system on a device, do

```
mkfs <device> <kb> <lvid>
```

The second argument is the number of kilobytes to give the volume for a quota, and the last argument is a volume identifier to assign to the resulting volume. You can choose you volume identifiers yourself, or you can use **newvid** to generate unique volume identifiers for you. **Rmfs**destroys**the**file**system**without the volume.

getroot, setroot: Establishing a root volume

When you start a server, it expects to be given a volume ID for the root of its file system. If no root is given, you can format devices, serve and unserve, and and make file systems, but you cannot use them in any way until you establish the root volume.

```
setroot <lvid>
```

establishes the root node on the given volume as the directory that corresponds to "/" in all file system activity that ensues.

```
getroot
```

returns the root if it has been set.

If your devices have already been formatted, and your file systems already made, you can start your server and run it in the background, by using the configuration/command-line option *svas_root* to establish the root volume.

mount, pmount, dismount: Establishing a root volume

The command

```
mount
```

simply prints the mount table (the table of file systems about which the server has information). Each time the file system is expanded by linking volumes together (below), the Shore server stores meta data about the link in an index at the root of the volume. This permits you to find all the persistent links without scanning the entire file system. The **mount** command gathers it information from these indexes. It indicates, with a **T** the root volume, and with a **P** those volumes that are the targets of persistent, cross-file system links.

To gain access to a file system other than your root file system, you can create links to other (already-formatted and served) volumes with:

```
pmount <lvid> <mountpoint>
```

The volume identifier indicates a volume already served by some Shore server. The mount point is a path in the name space visible to this server. The directory indicated by the path must exist, and be writable, and the last component of the path cannot name an object that exists. This command is similar to the Unix *ln* command; it makes a (hard) link to the root directory of the given volume. You must also have write permission for the root directory of the volume identified by <lvid>, because its entry ".." is changed.

```
dismount <mountpoint>
```

does the reverse of **pmount.** The directories at both ends of the link must be writable.

patch: fixing broken cross-volume links

It is possible for cross-volume (cross-file system) links to be corrupted, for example, when one of two devices is destroyed while the server is down. In some cases it is necessary to patch the links. This command lets you effectively undo these cross-file system links without considering corrupted states to be error states. It also cleans up the metadata that are stored on each volume to keep track of these persistent links.

VERSION

This manual page applies to Version 1.1 of the Shore software.

SPONSORSHIP

The Shore project is sponsored by the Advanced Research Project Agency, ARPA order number 018 (formerly 8230), monitored by the U.S. Army Research Laboratory under contract DAAB07-91-C-Q518.

SEE ALSO