NAME

restore, **rrestore** - restore files or file systems from backups made with dump

SYNOPSIS

restore -i [**-dDhmNuvy**] [**-b** *blocksize*] [**-f** *file* | **-P** *pipecommand*] [**-s** *fileno*]
restore -R [**-dDNuvy**] [**-b** *blocksize*] [**-f** *file* | **-P** *pipecommand*] [**-s** *fileno*]
restore -r [**-dDNuvy**] [**-b** *blocksize*] [**-f** *file* | **-P** *pipecommand*] [**-s** *fileno*]
restore -t [**-dDhNuvy**] [**-b** *blocksize*] [**-f** *file* | **-P** *pipecommand*] [**-s** *fileno*] [*file ...*]
restore -x [**-dDhmNuvy**] [**-b** *blocksize*] [**-f** *file* | **-P** *pipecommand*] [**-s** *fileno*] [*file ...*]

DESCRIPTION

The **restore** utility performs the inverse function of dump(8). A full backup of a file system may be restored and subsequent incremental backups layered on top of it. Single files and directory subtrees may be restored from full or partial backups. The **restore** utility works across a network; to do this see the **-f** and **-P** flags described below. Other arguments to the command are file or directory names specifying the files that are to be restored. Unless the **-h** flag is specified (see below), the appearance of a directory name refers to the files and (recursively) subdirectories of that directory.

restore may also be invoked as **rrestore**. The 4.3BSD option syntax is implemented for backward compatibility, but is not documented here.

Exactly one of the following flags is required:

-i This mode allows interactive restoration of files from a dump. After reading in the directory information from the dump, **restore** provides a shell like interface that allows the user to move around the directory tree selecting files to be extracted. The available commands are given below; for those commands that require an argument, the default is the current directory.

add [*arg*] The current directory or specified argument is added to the list of files to be extracted. If a directory is specified, then it and all its descendents are added to the extraction list (unless the **-h** flag is specified on the command line). Files that are on the extraction list are prepended with a ''*'' when they are listed by **ls**.

cd *arg* Change the current working directory to the specified argument.

delete [*arg*] The current directory or specified argument is deleted from the list of files to be extracted. If a directory is specified, then it and all its descendents are deleted from the extraction list (unless the **-h** flag is specified on the command line). The most expedient way to extract most of the files from a directory is to add the directory to the extraction list and then delete those files that are not needed.

extract All the files that are on the extraction list are extracted from the dump. The **restore** utility will ask which volume the user wishes to mount. The fastest way to extract a few files is to start with the last volume, and work towards the first volume.

help List a summary of the available commands.

ls [*arg*] List the current or specified directory. Entries that are directories are appended with a ''/''. Entries that have been marked for extraction are prepended with a ''*''. If the verbose flag is set the inode number of each entry is also listed.

pwd Print the full pathname of the current working directory.

quit Exit immediately, even if the extraction list is not empty.

setmodes All the directories that have been added to the extraction list have their owner, modes, and times set; nothing is extracted from the dump. This is useful for cleaning up after a restore has been prematurely aborted.

verbose The sense of the **-v** flag is toggled. When set, the verbose flag causes the **ls** command to list the inode numbers of all entries. It also causes **restore** to print out information about each file as it is extracted.

what Display dump header information, which includes: date, level, label, and the file system and host dump was made from.

-R Request a particular tape of a multi volume set on which to restart a full restore (see the **-r** flag below). This is useful if the restore has been interrupted.

-r Restore (rebuild a file system). The target file system should be made pristine with newfs(8), mounted and the user cd(1)'d into the pristine file system before starting the restoration of the initial level 0 backup. If the level 0 restores successfully, the **-r** flag may be used to restore any necessary incremental backups on top of the level 0. The **-r** flag precludes an interactive file extraction and can be detrimental to one's health if not used carefully (not to mention the disk). An example:

```
newfs /dev/da0s1a
mount /dev/da0s1a /mnt
cd /mnt
```

restore rf /dev/sa0

Note that **restore** leaves a file *restoresymtable* in the root directory to pass information between incremental restore passes. This file should be removed when the last incremental has been restored.

The **restore** utility , in conjunction with newfs(8) and dump(8), may be used to modify file system parameters such as size or block size.

-t The names of the specified files are listed if they occur on the backup. If no file argument is given, then the root directory is listed, which results in the entire content of the backup being listed, unless the **-h** flag has been specified. Note that the **-t** flag replaces the function of the old dumpdir(8) program.

-x The named files are read from the given media. If a named file matches a directory whose contents are on the backup and the **-h** flag is not specified, the directory is recursively extracted. The owner, modification time, and mode are restored (if possible). If no file argument is given, then the root directory is extracted, which results in the entire content of the backup being extracted, unless the **-h** flag has been specified.

The following additional options may be specified:

-b *blocksize*
The number of kilobytes per dump record. If the **-b** option is not specified, **restore** tries to determine the media block size dynamically.

-d Sends verbose debugging output to the standard error.

-D This puts **restore** into degraded mode, causing restore to operate less efficiently but to try harder to read corrupted backups.

-f *file* Read the backup from *file*; *file* may be a special device file like */dev/sa0* (a tape drive), */dev/da1c* (a disk drive), an ordinary file, or '**-**' (the standard input). If the name of the file is of the form "host:file", or "user@host:file", **restore** reads from the named file on the remote host using rmt(8).

-P *pipecommand*
Use popen(3) to execute the sh(1) script string defined by *pipecommand* as the input for every volume in the backup. This child pipeline's stdout (*/dev/fd/1*) is redirected to the **restore** input stream, and the environment variable RESTORE_VOLUME is set to the current volume

number being read. The *pipecommand* script is started each time a volume is loaded, as if it were a tape drive.

-h Extract the actual directory, rather than the files that it references. This prevents hierarchical restoration of complete subtrees from the dump.

-m Extract by inode numbers rather than by file name. This is useful if only a few files are being extracted, and one wants to avoid regenerating the complete pathname to the file.

-N Do the extraction normally, but do not actually write any changes to disk. This can be used to check the integrity of dump media or other test purposes.

-s *fileno*
Read from the specified *fileno* on a multi-file tape. File numbering starts at 1.

-u When creating certain types of files, restore may generate a warning diagnostic if they already exist in the target directory. To prevent this, the **-u** (unlink) flag causes restore to remove old entries before attempting to create new ones. This flag is recommended when using extended attributes to avoid improperly accumulating attributes on pre-existing files.

-v Normally **restore** does its work silently. The **-v** (verbose) flag causes it to type the name of each file it treats preceded by its file type.

-y Do not ask the user whether to abort the restore in the event of an error. Always try to skip over the bad block(s) and continue.

ENVIRONMENT

TAPE Device from which to read backup.

TMPDIR Name of directory where temporary files are to be created.

FILES

/dev/sa0	the default tape drive
*/tmp/rstdir**	file containing directories on the tape.
*/tmp/rstmode**	owner, mode, and time stamps for directories.
./restoresymtable	information passed between incremental restores.

DIAGNOSTICS

The **restore** utility complains if it gets a read error. If **-y** has been specified, or the user responds 'y', **restore** will attempt to continue the restore.

If a backup was made using more than one tape volume, **restore** will notify the user when it is time to mount the next volume. If the **-x** or **-i** flag has been specified, **restore** will also ask which volume the user wishes to mount. The fastest way to extract a few files is to start with the last volume, and work towards the first volume.

There are numerous consistency checks that can be listed by **restore**. Most checks are self-explanatory or can ''never happen''. Common errors are given below.

<filename>: not found on tape

The specified file name was listed in the tape directory, but was not found on the tape. This is caused by tape read errors while looking for the file, and from using a dump tape created on an active file system.

expected next file <inumber>, got <inumber>

A file that was not listed in the directory showed up. This can occur when using a dump created on an active file system.

Incremental dump too low

When doing incremental restore, a dump that was written before the previous incremental dump, or that has too low an incremental level has been loaded.

Incremental dump too high

When doing incremental restore, a dump that does not begin its coverage where the previous incremental dump left off, or that has too high an incremental level has been loaded.

Tape read error while restoring <filename>
Tape read error while skipping over inode <inumber>
Tape read error while trying to resynchronize

A tape (or other media) read error has occurred. If a file name is specified, then its contents are probably partially wrong. If an inode is being skipped or the tape is trying to resynchronize, then no extracted files have been corrupted, though files may not be found on the tape.

resync restore, skipped <num> blocks

After a dump read error, **restore** may have to resynchronize itself. This message lists the number of blocks that were skipped over.

SEE ALSO

dump(8), mount(8), newfs(8), rmt(8)

HISTORY

The **restore** utility appeared in 4.2BSD.

BUGS

The **restore** utility can get confused when doing incremental restores from dumps that were made on active file systems without the **-L** option (see dump(8)).

A level zero dump must be done after a full restore. Because restore runs in user code, it has no control over inode allocation; thus a full dump must be done to get a new set of directories reflecting the new inode numbering, even though the contents of the files is unchanged.

To do a network restore, you have to run restore as root. This is due to the previous security history of dump and restore. (restore is written to be setuid root, but we are not certain all bugs are gone from the restore code - run setuid at your own risk.)

The temporary files */tmp/rstdir** and */tmp/rstmode** are generated with a unique name based on the date of the dump and the process ID (see mktemp(3)), except for when **-r** or **-R** is used. Because **-R** allows you to restart a **-r** operation that may have been interrupted, the temporary files should be the same across different processes. In all other cases, the files are unique because it is possible to have two different dumps started at the same time, and separate operations should not conflict with each other.