Chapter 18 Filesystem Features: Attributes, Creating, Checking, Mounting - Notes

18.3 Learning Objectives:

- Explain concepts such as inodes, directory files, and extended attributes.
- Create and format filesystems.
- Check and fix errors on filesystems.
- Mount and unmount filesystems.
- Use netw ork file systems.
- Understand how to automount filesystems and make sure they mount at boot.

18.4 Extended Attributes and lsattr/chattr

Extended Attributes associate metadata not interpreted directly by filesystem w ith files. Four namespaces exist:

- user
- trusted
- security
- system

System namespace used for **Access Control Lists** (**ACL**s) and security namespace used by SELinux.

Flag values stored in file inode, maybe modified and set only by root user. View ed w ith **lsattr** and set w ith **chattr**. Flags may be set for files:

- `i` : Immutable. File w ith immutable attribute cannot be modified (not even my root). Cannot be deleted or renamed. No hard link can be created to it, and no data can be w ritten to file. Only superuser can set or clear this attribute.
- `a` : Append-only. FIle w ith append-only attribute set can only be opened in append mode for w riting. Only superuser can set ot clear this attribute.
- `d` : No-dump. File w ith no-dump attribute set is ignored w hen the **dump** program is run. Useful for sw ap and cache files that you don't w ant to w aste time backing up.
- `A` : No **atime** update. File w ith no-atime-update attribute set w ill not modify its **atime** (access time) record w hen file accessed but not otherw ise modified. Can increase performance on some systems because it reduces amount of disk I/O on system.

Note: there are other flags that can be set; typing `man chattr` w ill show w hole list. Format for **chattr**:

```
$ chattr [+|-|=mode] filename
```

lsattr used to display attributes for file:

```
$ lsattr filename
```

18.5 Creating and Formatting Filesystems

Every filesystem type has utility for **formatting** (making) filesystem on partition. Generic name for these utilities: **mkfs**. How ever, this is just frontend for filesystem-specific programs, each of w hich may have particular options.

mkfs

Thus, follow ing tw o commands entirely equivalent:

```
$ sudo mkfs -t ext4 /dev/sda10
$ sudo mkfs.ext4 /dev/sda10
```

General format for **mkfs**:

```
mkfs [-t fstype] [options] [device-file]
```

w here `[device-file]` is usually device name like `/dev/sda3` of `/dev/vg/lvm1`.

Each filesystem has ow n particular options that can be set w hen formatting. Eg. w hen creating **ext4** filesystem, journalling settings = one thing to keep in mind. Include defining journal file size, w hether or not to use external journal file.

Should look at **man** page for each of **mkfs.*** programs to see details.

18.6 Checking an Repairing Filesystems

Every filesystem type has utility designed to check for errors (and hopefully fix any that are found). Generic name for these utilities: **fsck**. How ever, just frontend for filesystem-specific programs.

fsck

Thus, follow ing tw o commands entirely equivalent:

```
$ sudo fsck -t ext4 /dev/sda10
$ sudo fsck.ext4 /dev/sda10
```

If filesystem is of type understood by operating system, can almost alw ays just do:

```
$ sudo fsck /dev/sda10
```

and system w ill figure out type by examining first few bytes on partition.

fsck run automatically after set number of mounts, or set interval since last time it w as run, or after abnormal shutdow n. Should only be run on unmounted filesystems. Can force a check of all ounted filesystems at boot by doing:

```
$ sudo touch /forcefsck
$ sudo reboot
```

The file `/forcefsck` w ill disappear after successful check. One reason this is valuable trick: can do **fsck** on root filesystem, w hich is hard to do on running system.

General format for **fsck**:

```
fsck [-t fstype] [options] [device-file]
```

w here `[device-file]` is usually device name like `/dev/sda3` or `/dev/vg/lvm1`. Usually, do not need to specify filesystem type, as **fsck** can figure it out by examining superblocks at start of partition.

Can control w hether any errors found should be fixed one by one manually w ith `-r` option, or automatically, as best possible, by using `-a` option, etc. In addition, each filesystem type may have ow n particular options that can be set w hen checking.

Note: **journalling** filesystems much faster to check than older generation filesystems for tw o reasons:

- Rarely need to scan entire partition for errors, as everything but very last transaction logged and confirmed, so takes almost no time to check
- Even if w hole filesystem checked, new er filesystems designed w ith **fast fsck** in mind. Older filesystems did not think much about this w hen designed as sizes w ere much smaller

Should look at **man** page for each of **fsck.*** programs to see details.

18.7 Mounting Filesystems

All accessible files in Linux organized into one large hierarchical tree structure w ith head of tree being root directory (`/`). How ever, common to have more than one partition (each of w hich can have ow n filesystem type) joined together in same filesystem tree. Partitions can also be on different physical devices, even on netw ork.

mount program allow s attaching at any point in tree structure. **umount** allow s detaching them.

Mount point is directory w here filesystem attached. Must exist before **mount** can use it. **mkdir** can be used to create expty directory. If pre-existing directory used + contains files prior to being used as mount point, files w ill be hidden after mounting. These files are *not* deleted and w ill again be visible w hen filesystem unmounted.

By default, only superuser can mount/unmount filesystems.

Each filesystem mounted under specific directory:

```
$ sudo mount -t ext4 /dev/sdb4 home
```

- Mounts **ext4** filesystem
- Usually not necessary to specify type w ith `-t` option
- Filesystem is located on specific partition of hard drive (`/dev/sdb4`)
- Filesystem mounted at position `/home` in current directory tree
- Any files residing in original `/home` directory hidden until partition unmounted

18.10 mount

General form for mount:

```
mount [options] <source> <directory>
```

Note: in this example, filesystem mounted by using device node it resides on. How ever, also possible to mount using label or **UUID**. Thus, follow ing all equivalent:

```
$ sudo mount /dev/sda2  /home
$ sudo mount LABEL=home /home
$ sudo mount    -L home /home
$ sudo mount UUID=26d58ee2-9d20-4dc7-b6ab-aa87c3cfb69a /home
$ sudo mount   -U 26d58ee2-9d20-4dc7-b6ab-aa87c3cfb69a /home
```

Labels assigned by filesystem type specific utilities, such as **e2label**, and **UUIDs** assigned when partitions created as containers for filesystem, formatted with **mkfs**.

While any of these three methods for specifying device can be used, modern systems deprecate using device node form because names can change according to how the system is booted, which hard drives found first, etc. Labels = improvement, but on rare occasions, could have two partitions that wind up with same label. **UUIDs**, however, should always be unique + created when partitions created.

18.11 mount Options

mount takes many options, some generic like `-a` (mount all filesystems mentioned in `/etc/fstab`) and many filesystem specific. Has very long **man** page. Common example:

```
$ sudo mount -o remount,ro /myfs
```

which remounts filesystem with **read-only** attribute.

An exampe of hot to get quick summary of mount options:

```
$ mount --help
```

mounthelp

18.12 umount

Filesystems can be unmounted:

```
$ umount [device-file | mount-point]
```

Below, some examples of how to unmount filesystem:

- Unmount `/home` filesystem:

```
  $ sudo umount /home
```

- Unmount the `/dev/sda3` device:

```
  $ sudo umount /dev/sda3
```

Note: command to unmount filesystem is **umount** (*not* **unmount**!).

Like **mount**, **umount** has many options, many of w hich specific to filesystem type. **man** pages = best source for specific option information.

Most common error encountered w hen unmounting filesystem: trying to do this on filesystem currently in use, ie. current applications using files or other entries in filesystem.

Can be as simple as having terminal w indow open in directory on mounted filesystem. Just using **cd** in that w indow, or killing it, w ill get rid of `device is busy` error and allow unmounting.

How ever, if other processes inducing this error, must kill those processes before unmounting filesystem. Can use **fuser** to find out w hich users using filesystem and kill them (be careful w ith this, may also w ant to w arn users first). Can also use **lsof** ("list open files") to try and see w hich files being used and blocking unmounting.

18.13 Network Shares (NFS)

Common to mount remote filesystems through netw ork shares so they appear as if they w ere on local machine. Probably most common method used historically: **NFS** (**N**etw ork **F**ile **S**ystem).

NFS originally developed by Sun Microsystems in 1989, has been continuously updated. Modern systems use **NFSv4**, w hich has been continuously updated since 2000.

Other netw ork filesystems include **AFS** (**A**ndrew **F**ile **S**ystem), and **SMB** (**S**erver **M**essage **B**ook), also termed **CIFS** (**C**ommon **I**nternet **F**ile **S**ystem).

Because netw ork filesystem may be unavailable at any time, either because it is not present on netw ork share, or the netw ork is unavailable, systems have to be prepared for this possibility.

Thus, in such circumstances, system should be instructed not to get hung, or blocked, w hile w aiting longer than specified period. Can be specified in **mount** command:

```
$ sudo mount -t nfs myserver.com:/shdir /mnt/shdir
```

or in `/etc/fstab`. Put follow ing line in `/etc/fstab` to mount on boot or w ith `mount -a`:

```
myserver.com:/shdir /mnt/shdir nfs rsize=8192,wsize=8192,timeo=14,intr 0 0
```

System may try to mount **NFS** filesystem before netw ork is up. `netdev` and `noauto` options can be used. For more information, check **man nfs**, examine **mount** options.

Can also be solved using **autofs** or **automount**.

Mount has large amount of options, some of w hich specific to **nfs**. See **man** pages for both **nfs** and **mount** for details.

18.14 Mounting at Boot

During system initialization, the command `mount -a` executed. Mounts all filesystems listed in `/etc/fstab` configuration file. Entries can refer to both local + remote netw ork-mounted filesystems. Below show s example of how to mount all filesystems listed in `/etc/fstab` configuration file during system boot.

File show s w hat filesystems may be automatically mounted at boot, and w here they may be found on the local machine or netw ork. Can specify w ho may mount them and w ith w hat permissions, and other relevant options. Some lines refer to special pseudo-filesystems such as **proc**, **sys**, **devpts**.

fstab

Each record in `/etc/fstab` file contains white space separated fields of information about filesystem to be mounted:

- Device node, label, or UUID

  For filesystems which do not have device node, such as **tmpfs**, **proc**, and **sysfs**, this field is just placeholder. Sometimes will see the word *none* in that column, or used on command line.

- Mount point

  Can also be placeholder, like for **swap**, which is not mounted anywhere.

- Filesystem type (ie. **ext4**, **xfs**, **btrfs**, **vfat**)

- A comma-separated list of options

- **dump** frequency (or a 0)

  Used by rarely used **`dump -w`** command.

- **fsck** pass number (or 0, meaning do not check state at boot)

mount and **umount** utilities can use information in `/etc/fstab`. In such case, could type:

```
$ sudo mount /usr/src
```

instead of

```
$ sudo mount LABEL=src /usr/src
```

in above example.

18.15 Automatic Filesystem Mounting

Linux systems have long had ability to mount filesystem only when filesystem needed. Historically, done using **autofs**. This utility requires installation of **autofs** package using appropriate package manager + configuration of files in `/etc`.

While **autofs** very flaxible + well understood, systemd-based systems (including all recent enterprise Linux distributions) come with **automount** facilities built into **systemd** framework. Configuring this = simple as adding a line in `/etc/fstab` specifying proper device, mount point, mount options:

```
LABEL=Sam128 /SAM ext4 noauto,x-systemd.automount,x-systemd.device-timeout=10,x-systemd.idle-timeout=30 0 0
```

and then, either rebooting or issuing command:

```
$ sudo systemctl daemon-reload
$ sudo systemctl restart local-fs.target
```

Next, will give example and explain options.

18.16 automount Example

Example provided next mounts USB pen drive that is always plugged into system, only when it is used. Options in `/etc/fstab`:

- `noauto`: do not mount at boot. Here, `auto` does not refer to **automount**
- `x-systemd.automount`: use the **systemd automount** facility
- `x-systemd.automount.device-timeout=10`: if this device is not available, say it is a network device accessible through NFS, timeout after 10 seconds instead of getting hung
- `x-systemd.automount.idle-timeout=30`: if device is not used for 30 seconds, unmount it

Note: device **may** be mounted during boot, but then should be unmounted after timeout specified. Below shows how device is only available one it is used.

automount

18.17 Listing Currently Mounted Filesystems

List of currently mounted filesystems can be seen by typing:

mountout

##

Back to top