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 network 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 with 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. Viewed with **lsattr** and set with **chattr**. Flags may be set for files:

- `i` : Immutable. File with 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 written to file. Only superuser can set or clear this attribute.
- `a` : Append-only. FIle with append-only attribute set can only be opened in append mode for writing. Only superuser can set ot clear this attribute.
- `d` : No-dump. File with no-dump attribute set is ignored when the **dump** program is run. Useful for swap and cache files that you don't want to waste time backing up.
- `A` : No **atime** update. File with no-atime-update attribute set will not modify its **atime** (access time) record when file accessed but not otherwise 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` will show whole 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**. However, this is just frontend for filesystem-specific programs, each of which may have particular options.

```
File  Edit  View  Search  Terminal  Help
c7:/tmp>ls -lh /sbin/mkfs*
-rwxr-xr-x 1 root root  12K Feb  8 04:36 /sbin/mkfs
-rwxr-xr-x 1 root root 301K Mar 16  2016 /sbin/mkfs.btrfs
-rwxr-xr-x 1 root root  33K Feb  8 04:36 /sbin/mkfs.cramfs
-rwxr-xr-x 4 root root  94K Jun 13  2016 /sbin/mkfs.ext2
-rwxr-xr-x 4 root root  94K Jun 13  2016 /sbin/mkfs.ext3
-rwxr-xr-x 4 root root  94K Jun 13  2016 /sbin/mkfs.ext4
-rwxr-xr-x 1 root root  28K Mar  4  2014 /sbin/mkfs.fat
-rwxr-xr-x 1 root root  33K Feb  8 04:36 /sbin/mkfs.minix
lrwxrwxrwx 1 root root    8 Apr  9  2015 /sbin/mkfs.msdos -> mkfs.fat
lrwxrwxrwx 1 root root   16 Nov 19  2016 /sbin/mkfs.ntfs -> /usr/sbin/mkntfs
lrwxrwxrwx 1 root root    8 Apr  9  2015 /sbin/mkfs.vfat -> mkfs.fat
-rwxr-xr-x 1 root root 360K Dec 22 10:37 /sbin/mkfs.xfs
c7:/tmp>
```

Thus, following two commands entirely equivalent:

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

General format for **mkfs**:

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

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

Each filesystem has own particular options that can be set when formatting. Eg. when creating **ext4** filesystem, journalling settings = one thing to keep in mind. Include defining journal file size, whether 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**. However, just frontend for filesystem-specific programs.

```
File  Edit  View  Search  Terminal  Help
c7:/tmp>ls -lh /sbin/fsck*
-rwxr-xr-x 1 root root  33K Feb  8 04:36 /sbin/fsck
-rwxr-xr-x 1 root root 1.2K Mar 16  2016 /sbin/fsck.btrfs
-rwxr-xr-x 1 root root  20K Feb  8 04:36 /sbin/fsck.cramfs
-rwxr-xr-x 4 root root 251K Jun 13  2016 /sbin/fsck.ext2
-rwxr-xr-x 4 root root 251K Jun 13  2016 /sbin/fsck.ext3
-rwxr-xr-x 4 root root 251K Jun 13  2016 /sbin/fsck.ext4
-rwxr-xr-x 1 root root  57K Mar  4  2014 /sbin/fsck.fat
-rwxr-xr-x 1 root root  74K Feb  8 04:36 /sbin/fsck.minix
lrwxrwxrwx 1 root root    8 Apr  9  2015 /sbin/fsck.msdos -> fsck.fat
lrwxrwxrwx 1 root root   13 Nov 19  2016 /sbin/fsck.ntfs -> ../bin/ntfsck
lrwxrwxrwx 1 root root    8 Apr  9  2015 /sbin/fsck.vfat -> fsck.fat
-rwxr-xr-x 1 root root  433 Dec 22 10:37 /sbin/fsck.xfs
c7:/tmp>
```

Thus, following two 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 always just do:

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

and system will 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 was run, or after abnormal shutdown. 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` will disappear after successful check. One reason this is valuable trick: can do **fsck** on root filesystem, which is hard to do on running system.

General format for **fsck**:

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

where `[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 whether any errors found should be fixed one by one manually with `-r` option, or automatically, as best possible, by using `-a` option, etc. In addition, each filesystem type may have own particular options that can be set when checking.

Note: **journalling** filesystems much faster to check than older generation filesystems for two 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 whole filesystem checked, newer filesystems designed with **fast fsck** in mind. Older filesystems did not think much about this when designed as sizes were 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 with head of tree being root directory (`/`). However, common to have more than one partition (each of which can have own filesystem type) joined together in same filesystem tree. Partitions can also be on different physical devices, even on network.

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

Mount point is directory where 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 will be hidden after mounting. These files are *not* deleted and will again be visible when 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 with `-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. However, also possible to mount using label or **UUID**. Thus, following 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
```

```
 File  Edit  View  Search  Terminal  Help
c:/tmp> mount --help
Usage:
 mount [-lhV]
 mount -a [options]
 mount [options] [--source] <source> | [--target] <directory>
 mount [options] <source> <directory>
 mount <operation> <mountpoint> [<target>]

Options:
 -a, --all              mount all filesystems mentioned in fstab
 -c, --no-canonicalize  don't canonicalize paths
 -f, --fake             dry run; skip the mount(2) syscall
 -F, --fork             fork off for each device (use with -a)
 -T, --fstab <path>     alternative file to /etc/fstab
 -h, --help             display this help text and exit
 -i, --internal-only    don't call the mount.<type> helpers
 -l, --show-labels      lists all mounts with LABELs
 -n, --no-mtab          don't write to /etc/mtab
 -o, --options <list>   comma-separated list of mount options
 -O, --test-opts <list> limit the set of filesystems (use with -a)
 -r, --read-only        mount the filesystem read-only (same as -o ro)
 -t, --types <list>     limit the set of filesystem types
     --source <src>     explicitly specifies source (path, label, uuid)
     --target <target>  explicitly specifies mountpoint
 -v, --verbose          say what is being done
 -V, --version          display version information and exit
 -w, --rw, --read-write mount the filesystem read-write (default)

 -h, --help     display this help and exit
 -V, --version  output version information and exit
....
c7:/tmp>
```

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 which specific to filesystem type. **man** pages = best source for specific option

information.

Most common error encountered when 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 window open in directory on mounted filesystem. Just using **cd** in that window, or killing it, will get rid of `device is busy` error and allow unmounting.

However, if other processes inducing this error, must kill those processes before unmounting filesystem. Can use **fuser** to find out which users using filesystem and kill them (be careful with this, may also want to warn users first). Can also use **lsof** ("list open files") to try and see which files being used and blocking unmounting.

18.13 Network Shares (NFS)

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

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

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

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

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

or in `/etc/fstab` . Put following line in `/etc/fstab` to mount on boot or with `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 network 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 which 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 network-mounted filesystems. Below shows example of how to mount all filesystems listed in `/etc/fstab` configuration file during system boot.

File shows what filesystems may be automatically mounted at boot, and where they may be found on the local machine or network. Can specify who may mount them and with what permissions, and other relevant options. Some lines refer to special pseudo-filesystems such as **proc**, **sys**, **devpts**.

```
File  Edit  View  Search  Terminal  Help
c7:/tmp>cat /etc/fstab
#
# /etc/fstab
# Created by anaconda on Thu Jan 15 19:25:00 2015
#
# Accessible filesystems, by reference, are maintained under '/dev/disk'
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info
#
LABEL=RHEL7        /              ext4       defaults        1 1
LABEL=local      /usr/local    ext4   defaults 1 2
LABEL=src        /usr/src      ext4   defaults 1 2
LABEL=pictures   /PICTURES     ext4   defaults 1 2
LABEL=dead       /DEAD         ext4   defaults 1 2
LABEL=dead2      /DEAD2        ext4   defaults 1 2
LABEL=virtual    /VIRTUAL      ext4   defaults 1 2
LABEL=iso_images /ISO_IMAGES ext4   defaults 1 2
LABEL=audio      /AUDIO        ext4   defaults 1 2
LABEL=vms        /VMS           ext4  defaults 1 2
/usr/src/KERNELS.sqfs /usr/src/KERNELS squashfs loop 0 0

LABEL=SWAP swap                     swap      defaults        0 0
#UUID=471dfeba-3ec7-4529-8069-2afe50762c57 / ext4   defaults  1 1
c7:/tmp>
```

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 flexible + 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
```
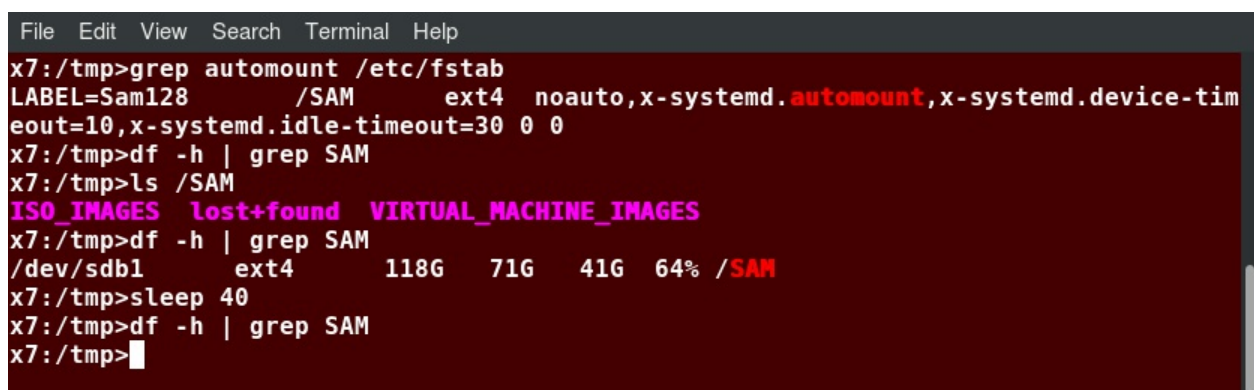
Next, will give example and explain options.

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.

18.17 Listing Currently Mounted Filesystems

List of currently mounted filesystems can be seen by typing:

```
c7:/tmp> mount

sysfs on /sys type sysfs (rw,nosuid,nodev,noexec,relatime)
proc on /proc type proc (rw,nosuid,nodev,noexec,relatime)
devtmpfs on /dev type devtmpfs (rw,nosuid,size=8126336k,nr_inodes=2031584,mode=755)
securityfs on /sys/kernel/security type securityfs (rw,nosuid,nodev,noexec,relatime)
devpts on /dev/pts type devpts (rw,nosuid,noexec,relatime,gid=5,mode=620,ptmxmode=000)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev)
tmpfs on /run type tmpfs (rw,nosuid,nodev,mode=755)
tmpfs on /run/user/1000 type tmpfs (rw,nosuid,nodev,relatime,size=1628296k,mode=700,uid=1000,gid=1000)
tmpfs on /sys/fs/cgroup type tmpfs (ro,nosuid,nodev,noexec,mode=755)
cgroup on /sys/fs/cgroup/systemd type cgroup (rw,nosuid,nodev,noexec,relatime,xattr,release_agent=/usr/lib/syst
emd/systemd-cgroups-agent,name=systemd)
cgroup on /sys/fs/cgroup/perf_event type cgroup (rw,nosuid,nodev,noexec,relatime,perf_event)
....
configfs on /sys/kernel/config type configfs (rw,relatime)
/dev/sdb1 on / type ext4 (rw,relatime,data=ordered)
systemd-1 on /proc/sys/fs/binfmt_misc type autofs (rw,relatime,fd=34,pgrp=1,timeout=300,minproto=5,maxproto=5,d
irect)
debugfs on /sys/kernel/debug type debugfs (rw,relatime)
mqueue on /dev/mqueue type mqueue (rw,relatime)
hugetlbfs on /dev/hugepages type hugetlbfs (rw,relatime)
nfsd on /proc/fs/nfsd type nfsd (rw,relatime)
binfmt_misc on /proc/sys/fs/binfmt_misc type binfmt_misc (rw,relatime)
/dev/mapper/VG-src on /usr/src type ext4 (rw,relatime,stripe=32667,data=ordered)
/dev/mapper/VG-local on /usr/local type ext4 (rw,relatime,data=ordered)
/dev/mapper/VG-vms on /VMS type ext4 (rw,relatime,stripe=32719,data=ordered)
/dev/mapper/VG2-dead on /DEAD type ext4 (rw,relatime,stripe=32717,data=ordered)
.....
/dev/mapper/VG2-audio on /AUDIO type ext4 (rw,relatime,data=ordered)
/usr/src/KERNELS.sqfs on /usr/src/KERNELS type squashfs (ro,relatime)
sunrpc on /var/lib/nfs/rpc_pipefs type rpc_pipefs (rw,relatime)
fusectl on /sys/fs/fuse/connections type fusectl (rw,relatime)
vmware-vmblock on /run/vmblock-fuse type fuse.vmware-vmblock (rw,nosuid,nodev,relatime,user_id=0,group_id=0,def
ault_permissions,allow_other)
gvfsd-fuse on /run/user/1000/gvfs type fuse.gvfsd-fuse (rw,nosuid,nodev,relatime,user_id=1000,group_id=1000)
c7:/tmp>
```

##

Back to top