

System Administration (2)

IT401

Prepared by
Dr. Ahmed El-Douh

Lecture 8



System Administration 2

Lecture 8

IT401

Access Linux File Systems

Prepared by
Dr. Ahmed El-Douh

Objectives

Upon completion of this lecture, you will be able to:

- Identify the file system hierarchy.
- Access and view the contents of file systems.
- Use hard links and symbolic links to create multiple names for the same file.
- Search for files on mounted file systems.

Outlines

- Identifying File Systems and Devices
- Mounting and Unmounting File Systems
- Making Links Between Files
- Locating Files on the System



Identifying File Systems and Devices



Storage Management Concepts

- A **file system** is an organized structure that stores files and directories on a storage device such as a disk or partition.
- The **file system hierarchy** combines all file systems into one directory tree with a single root: `/`.
- One key advantage of this design is that storage can be expanded at any time by adding a new disk or partition that contains a supported file system.
- Adding a new file system to the existing hierarchy is called **mounting**.
- The directory where the new file system becomes accessible is called the **mount point**.
- This is different from Microsoft Windows, where each new file system appears as a separate **drive letter**.

Storage Devices and Partitions

- Hard disks and storage devices are divided into smaller units called **partitions**.
- A **partition** allows you to organize a disk into separate sections.
Each section can have a different **file system** or a different purpose.
- Example:
 - One partition can store **user home directories**.
 - Another partition can store **system data and logs**.
- If one partition becomes full, other partitions remain usable.
This helps in planning and managing storage more efficiently.

Block Devices and Device Names

- Storage devices in Linux are represented as **block devices**. These appear under the **/dev** directory.
- Common device names:
 - The first detected disk → **/dev/sda**
 - The second disk → **/dev/sdb**
 - The third → **/dev/sdc**
 - ...and so on.
- Each disk is the whole device.

Partitions on that disk are numbered:

- First partition: **/dev/sda1**
- Second partition: **/dev/sda2**
- Third partition: **/dev/sda3**

```
[student@serverX ~]$ ls -l /dev/vda
brw-rw----. 1 root disk 253, 0 Mar 13 08:00 / dev/vda
```

Examining File Systems

- To view mounted file systems and check available and used disk space, use the **df** command.
- Running **df** without any arguments displays:
 - Total disk space
 - Used space
 - Available space
 - Percentage of disk usage
 - Mount points of all active file systems
- The command reports information for both local and remote file systems.

```
[student@serverX ~]$ df
Filesystem      1K-blocks    Used Available Use% Mounted on
/dev/vda1        6240256 4003760   2236496   65% /
devtmpfs          950536         0    950536    0% /dev
tmpfs            959268         80    959188    1% /dev/shm
tmpfs            959268        2156    957112    1% /run
tmpfs            959268         0    959268    0% /sys/fs/cgroup
```

Examining File Systems

Show a report on the file systems on the serverX machine with all units converted to human-readable format:

```
[student@serverX ~]$ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/vda1        6.0G  3.9G  2.2G  65% /
devtmpfs         929M    0  929M   0% /dev
tmpfs            937M   80K  937M   1% /dev/shm
tmpfs            937M  2.2M  935M   1% /run
tmpfs            937M    0  937M   0% /sys/fs/cgroup
```

Examining File Systems

Show a disk usage report for the **/root** directory on serverX:

```
[root@serverX ~]# du /root
4 /root/.ssh
4 /root/.cache/dconf
4 /root/.cache
4 /root/.dbus/session-bus
4 /root/.dbus
```

Examining File Systems

Show a disk usage report in human-readable format for the **/var/log** directory on serverX:

```
[root@serverX ~]# du -h /var/log
...
4.9M /var/log/sa
68K /var/log/prelink
0 /var/log/qemu-ga
14M /var/log
```



Mounting and Unmounting File Systems



Mounting and Unmounting File Systems

Objectives

- After completing this section, students should be able to access the contents of file systems by adding file systems to the directory hierarchy through mounting.
- Students should also be able to remove file systems from the directory hierarchy through unmounting.

Mounting File Systems Manually

❖ Mounting File Systems Manually

- A file system located on a SATA, PATA, or SCSI device must be mounted manually in order to access it.
- The mount command allows the root user to manually mount a file system.
- The first argument to the mount command specifies the file system to mount.
- The second argument specifies the directory where the file system will be available after mounting.
- This directory is called the mount point.

❖ The mount command accepts the file system argument in two ways

- The device file of the partition holding the file system, located under **/dev**
- The **UUID**, which is the universal unique identifier of the file system

Mounting and Unmounting File Systems

Using **blkid**

- The **blkid** command provides information about existing partitions that contain a file system.
- It displays the UUID of each file system and the file system type used to format it.

Example Output

```
[root@serverX ~]# blkid  
/dev/vda1: UUID="46f543fd-78c9-4526-a857-244811be2d88" TYPE="xfs"
```

Mounting and Unmounting File Systems

Mount by device file of the partition that holds the file system.

```
[root@serverX ~]#  
mount /dev/vdb1 /mnt/mydata
```

Mount the file system by universal unique id, or the UUID, of the file system.

```
[root@serverX ~]#  
mount UUID="46f543fd-78c9-4526-a857-244811be2d88" /mnt/mydata
```

Unmounting File Systems

To unmount a file system, the **umount** command expects the mount point as an argument.

Change to the **/mnt/mydata** directory. Try to unmount the device mounted on the **/mnt/mydata** mount point. It will fail.

```
[root@serverX ~]# cd /mnt/mydata[root@serverX mydata]# umount /mnt/mydata
umount: /mnt/mydata: target is busy.
```

(In some cases useful info about processes that use
the device is found by `lsof(8)` or `fuser(1)`)

Mounting and Unmounting File Systems

- A file system cannot be unmounted if a process is accessing the mount point.
- **To successfully unmount**, the process must stop using the mount point.
- The **lsuf** command lists open files and the processes that are using them.
- This helps identify which process is preventing the unmount operation.

```
[root@serverX mydata]# lsuf /mnt/mydata
```

COMMAND	PID	USER	FD	TYPE	DEVICE	SIZE/OFF	NODE	NAME
bash	1593	root	cwd	DIR	253,2	6	128	/mnt/mydata
lsuf	2532	root	cwd	DIR	253,2	19	128	/mnt/mydata
lsuf	2533	root	cwd	DIR	253,2	19	128	/mnt/mydata



Making Links Between Files



Making Links Between Files

Objectives

- After completing this section, students should be able to create hard links and soft links.
- Students should understand how multiple names can point to the same file.

Making Links Between Files

Creating Hard Links

- A hard link is a new directory entry that points to the same file on the file system.
- Every file starts with one hard link by default.
- Creating a hard link allows multiple file names to refer to the same content without copying the data.
- Hard links must be created in the same file system.
- All hard links share the same permissions, ownership, timestamps, and content.
- The `ls -l` command shows the hard link count next to the permissions column.

```
[root@serverX ~]# echo "Hello World" > newfile.txt [root@serverX ~]# ls -l  
newfile.txt  
-rw-r--r--. 1 root root 0 Mar 11 19:19 newfile.txt
```

Making Links Between Files

Creating Hard Links with **ln**

- The **ln** command is used to **create new hard links to an existing file**.
- The first argument is the existing file.
- The second argument is the name or location of the new hard link.
- Hard links can be placed anywhere as long as they are on the same file system.
- Once created, all hard links are identical and there is no way to know which one was the original name.

Example

Create a hard link **newfile-link2.txt** for the existing file **newfile.txt** in the **/tmp** directory.

```
[root@serverX ~]# ln newfile.txt /tmp/newfile-hlink2.txt [root@serverX ~]# ls -l  
newfile.txt /tmp/newfile-hlink2.txt  
-rw-rw-r--. 2 root root 12 Mar 11 19:19 newfile.txt  
-rw-rw-r--. 2 root root 12 Mar 11 19:19 newfile-hlink2.txt
```

Making Links Between Files

Hard Links and File Deletion

- When the original file name is removed, the data itself remains available as long as at least one hard link still exists.
- The file content is not deleted until the last hard link is removed.

Example

```
[root@serverX ~]# rm -f newfile.txt
[root@serverX ~]# ls -l /tmp/newfile-link2.txt
-rw-rw-r--. 1 root root 12 Mar 11 19:19 /tmp/newfile-link2.txt
[root@serverX ~]# cat /tmp/newfile-link2.txt
Hello World
```

Making Links Between Files

Creating Soft Links

- The `ln -s` command creates a soft link, also called a symbolic link.
- A soft link is a special file that points to another file or directory.
- Unlike hard links, soft links can point to directories and can point to targets on different file systems.
- A soft link stores only the path of the original target.

```
[root@serverX ~]# ln -s /root/newfile-link2.txt /tmp/newfile-symlink.txt
[root@serverX ~]# ls -l newfile-link2.txt /tmp/newfile-symlink.txt
lrwxrwxrwx. 1 root root 11 Mar 11 20:59 /tmp/newfile-symlink.txt -> /root/newfile-link2.txt
-rw-rw-r--. 1 root root 12 Mar 11 19:19 newfile-link2.txt
```

- When the original file is deleted, the soft link still exists but points to a missing file.
- This is called a dangling soft link.

```
[root@serverX ~]# rm -f newfile-link2.txt
[root@serverX ~]# ls -l /tmp/newfile-symlink.txt
lrwxrwxrwx. 1 root root 11 Mar 11 20:59 /tmp/newfile-symlink.txt -> newfile-link2.txt
[root@serverX ~]# cat /tmp/newfile-symlink.txt
cat: /tmp/newfile-symlink.txt: No such file or directory
```

Making Links Between Files

Soft Links to Directories

- A soft link can point to a directory.
- When accessing the soft link with `cd`, it behaves exactly like entering the original directory.

Example

```
[root@serverX ~]# ln -s /etc /root/configfiles [root@serverX ~]# cd /root/  
configfiles [root@serverX configfiles]# pwd  
/root/configfiles
```



Locating Files on the System



Locating Files on the System

Objectives

- After completing this section, students should be able to search for files on mounted file systems using the `find` and `locate` commands.

Tools for Finding Files

A system administrator needs tools for searching files based on specific criteria within the file system.

This section covers two commands for searching files:

- **locate** searches a pre-generated database for file names or paths and returns results instantly.
- **find** searches the file system directly in real time by crawling through directories and matching specific conditions.

Locating Files on the System

Locating Files by Name with locate

- The **locate** command searches for files by name or path using a local database.
- The database contains a list of file names and paths.
- Results are returned only from directories where the user has read permissions.
- Use locate to quickly find files that match a specific keyword.

```
[student@serverX ~]$ locate passwd
/etc/passwd
/etc/passwd-
/etc/pam.d/passwd
/etc/security/opasswd
/usr/bin/gpasswd
/usr/bin/grub2-mkpasswd-pbkdf2
/usr/bin/lpasswd
/usr/bin/passwd
/usr/bin/userpasswd
/usr/bin/vino-passwd
/usr/bin/vncpasswd
```

Locating Files on the System

Partial matches with **locate**

- The **locate** command returns results even when the search keyword matches only part of the file name or directory path.
- When a user searches for a term, any file or directory that contains this term in its name or in any part of its path will be listed.

Example:

```
[root@serverX ~]# locate image  
/home/myuser/boot.image  
/home/someuser/my_family_image.png  
/home/student/myimages-vacation/picture.png
```

The **-i** option performs a case-insensitive search. With this option, all possible combinations of upper- and lowercase letters match the search.

```
[student@serverX ~]$ locate -i messages  
...
```

Locating Files on the System

Limiting results with the -n option

- The **-n** option limits the number of search results returned by the **locate** command.
- This is useful when the search keyword produces a large number of matches.
- For example, the following command returns only the first **five** matches for the file name **snow.png**:

```
[student@serverX ~]$ locate -n 5 snow.png
/usr/share/icons/HighContrast/16x16/status/weather-snow.png
/usr/share/icons/HighContrast/22x22/status/weather-snow.png
/usr/share/icons/HighContrast/24x24/status/weather-snow.png
/usr/share/icons/HighContrast/256x256/status/weather-snow.png
/usr/share/icons/HighContrast/32x32/status/weather-snow.png
```

Locating Files on the System

Searching for Files with `find`

- The `find` command performs a real-time search in the local file system.
- It searches for files that match the criteria provided in the command-line arguments.
- The command runs using the permissions of the current user.
- The user running `find` must have read and execute permissions on directories to see their contents.
- The first argument to `find` is the starting directory of the search.
- If no directory is provided, `find` starts searching in the current directory and all its subdirectories.
- To search the home directory of user `student`, start the search at: `/home/student`
- To search the entire system, start the search at: `/`

Locating Files on the System

Searching with the **-name** option

- The name option allows find to look for files that match a specific file name.
- It returns only the exact matches of the name provided.
- To search for a file named **sshd_config** starting from the root directory and all subdirectories, use:

```
[root@serverX ~]# find / -name sshd_config
```

- The name option can also match patterns when using wildcard characters.
- To search for all files ending with **.txt** starting from the root directory, use:

```
[root@serverX ~]# find / -name '*.txt'
```

Locating Files on the System

To search for files in **/etc/** that contain **pass** anywhere in their names on serverX, run:

```
[root@serverX ~]# find /etc -name '*pass*'
/etc/passwd
/etc/passwd-
/etc/fonts/conf.d/60-overpass.conf
/etc/selinux/targeted/modules/active/modules/passenger.pp
/etc/security/opasswd
/etc/pam.d/passwd
/etc/pam.d/password-auth-ac
/etc/pam.d/password-auth
/etc/pam.d/gdm-password
```

Locating Files on the System

find can search for files based on their ownership or permissions. Useful options when searching by owner are **-user** and **-group**, which search by name, and **-uid** and **-gid**, which search by ID.

Search for files owned by the user *student* in the **/home/student** directory on serverX.

```
[student@serverX ~]$ find -user student
.
./bash_logout
./bash_profile
./bashrc
./ssh
...
```

Locating Files on the System

Search for files owned by the group *student* in the **/home/student** directory on serverX.

```
[student@serverX ~]$ find -group student
.
./bash_logout
./bash_profile
./bashrc
./ssh
...
```

Locating Files on the System

Search for files owned by user ID *1000* in the **/home/student** directory on serverX.

```
[student@serverX ~]$ find -uid 1000
.
./bash_logout
./bash_profile
./bashrc
./ssh
...
```

Locating Files on the System

Search for files owned by group ID *1000* in the **/home/student** directory on serverX.

```
[student@serverX ~]$ find -gid 1000
.
./bash_logout
./bash_profile
./bashrc
./ssh
...
```

Locating Files on the System

Search for files owned by user *root* and group *mail* on the serverX machine.

```
[root@serverX ~]# find / -user root -group mail  
/var/spool/mail  
/var/spool/mail/root
```

Summary

- ✓ Identified file systems and storage devices using commands like **blkid**, **lsblk**, and device paths.
- ✓ **Mounted** and **unmounted** file systems using device names and UUIDs and handled mount point issues.
- ✓ Created and managed **hard links** and **soft links** and understood behavior, usage, and differences.
- ✓ Searched for files using **locate** and **find** based on name, type, ownership, patterns, and permissions.



Thank You