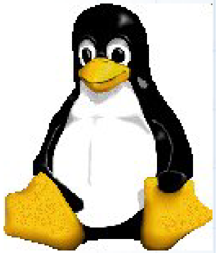
Working with files and directories



Unit objectives

After completing this unit, you should be able to:

- Describe the different file types
- Describe file and path names
- Create, delete, copy, move, and list directories
- Create, delete, copy, and move files
- View the content of both text and binary files

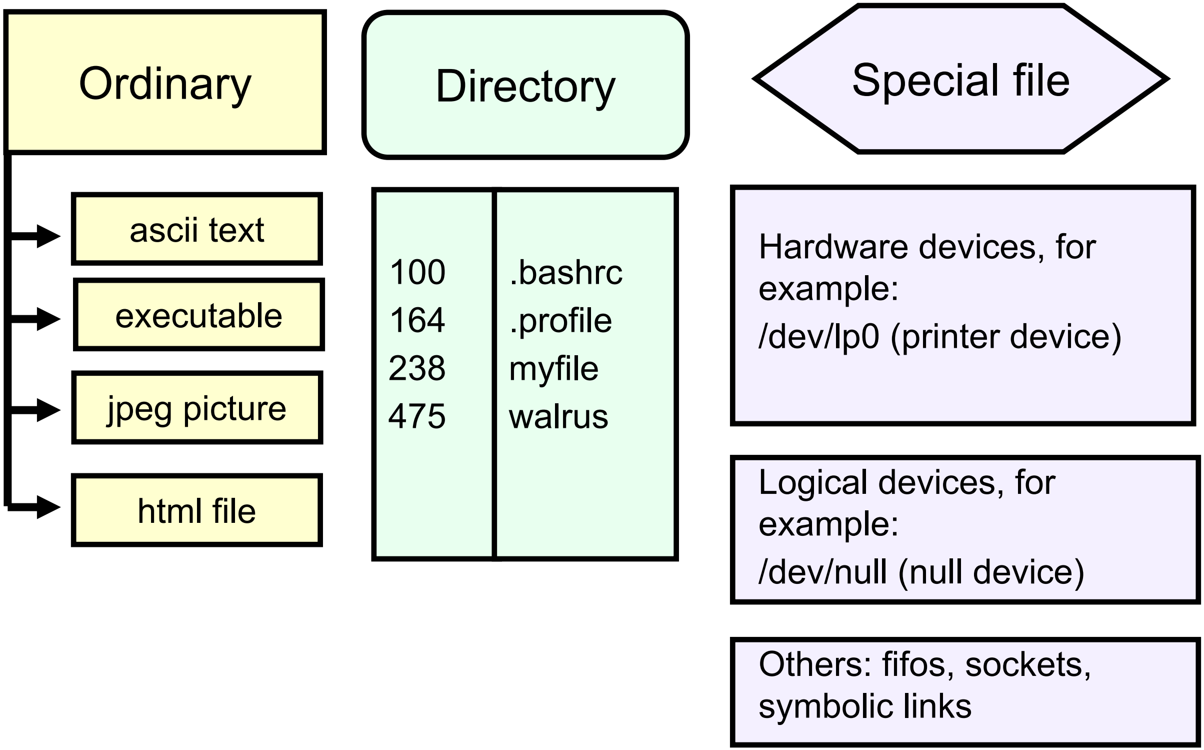
A file

- A file is:

```
A_collection_of_data\n
A_stream_of_characters_or_a_"byte_stream"\n
No_structure_is_imposed_on_a_file_by_the_operating_system\n
```

- \n is a newline character.
- _ is a space character.

File types

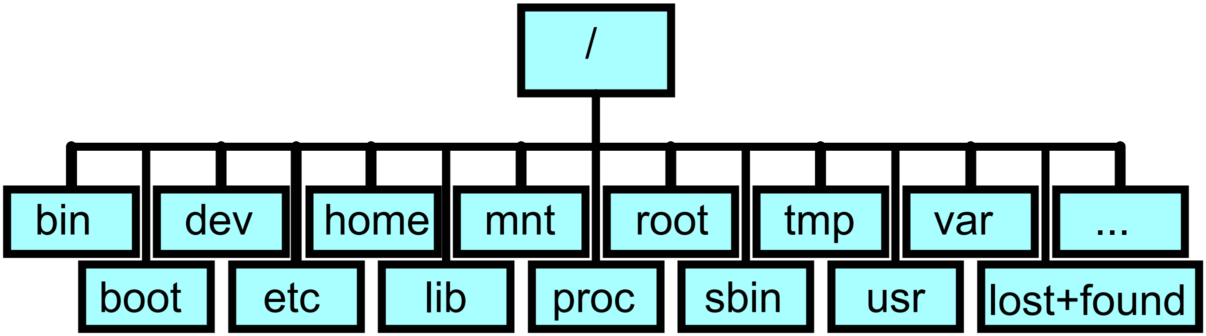

Ordinary
ascii text
executable
jpeg picture
html file
Directory
100 .bashrc
164 .profile
238 myfile
475 walrus
Special file
Hardware devices, for example:
/dev/lp0 (printer device)
Logical devices, for example:
/dev/null (null device)
Others: fifos, sockets, symbolic links

Linux file names

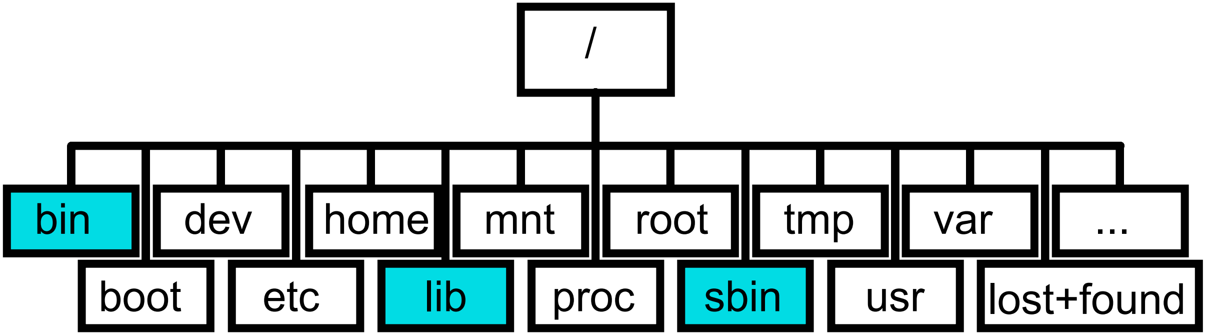
- Should be descriptive of the content
- Should use only alphanumeric characters
  - Uppercase, lowercase, number, @, _
- Should not include embedded blanks
- Should not contain shell metacharacters
  - * ? > < / ; & ! | \ ` ' " [ ] ( ) { }
- Should not begin with plus sign (+) or minus sign (-)
- Are case sensitive
- Are hidden if the first character is a period (.)
- Can have a maximum of 255 characters

Directory structure

- All Linux directories are contained in one virtual, unified file system.
- Physical devices are mounted on mount points.
  - USB flash drives
  - Hard disk partitions
  - Optical disk drives
- There are no drive letters like A:, C:, and so on.

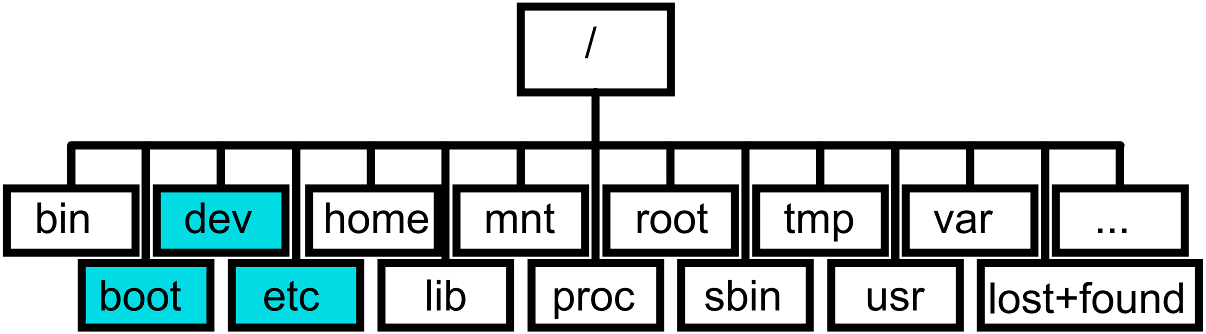


/bin, /lib, and /sbin



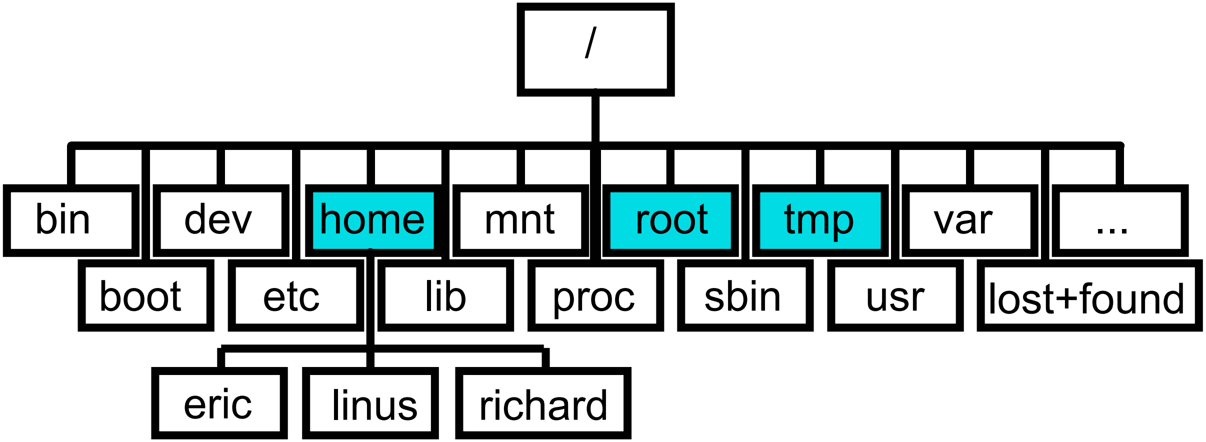
- `/bin` contains executables for every user.
- `/sbin` contains system administration executables.
- `/lib` contains libraries.
- They should always be available:
  - At system boot
  - In single user mode
  - When booting from rescue disk

/boot, /dev, and /etc



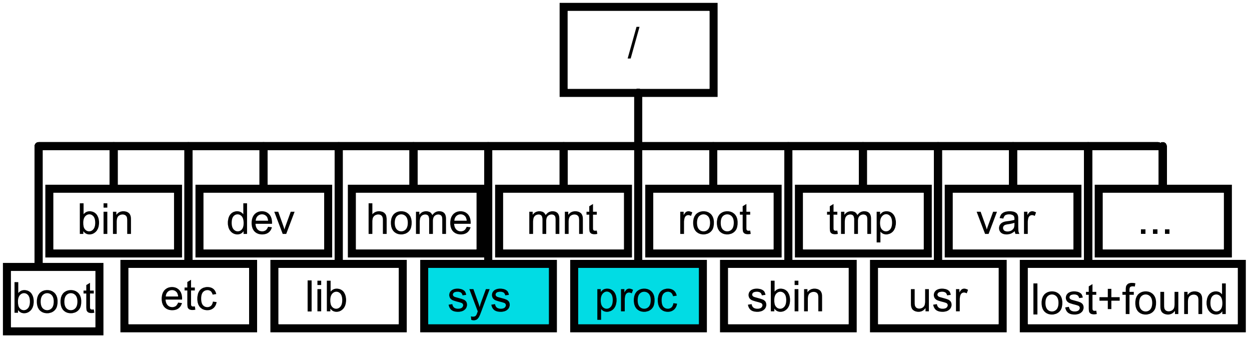
- Contains kernel image and some other goodies

/home, /root, and /tmp



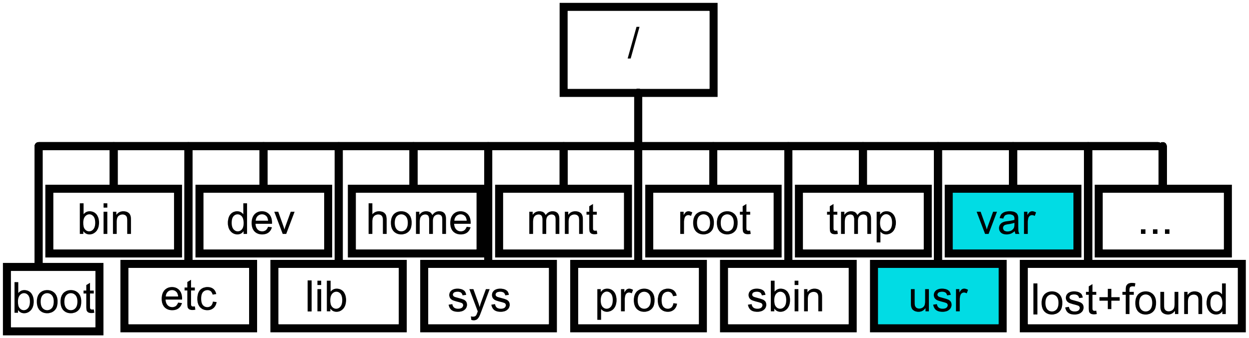
- These are the home directories of users.

/proc and /sys



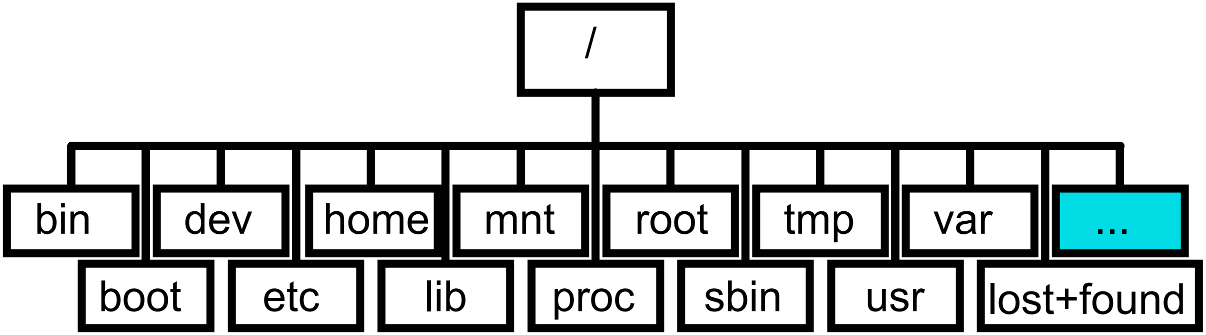
- These are virtual file systems.
- `/proc` represents kernel and process information.
- `/sys` represents driver and file system information.

/usr and /var



- `/usr` contains all the user programs that the system needs.
- `/var` contains data that is changed when the system is running normally. It is specific for each system, that is, not shared over the network with other computers.
- Logs often reside on `/var`, and they can get voluminous.

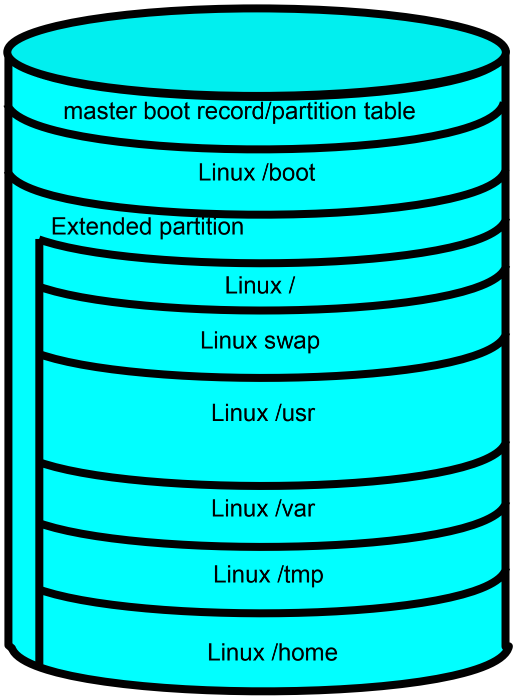
Other directories in /



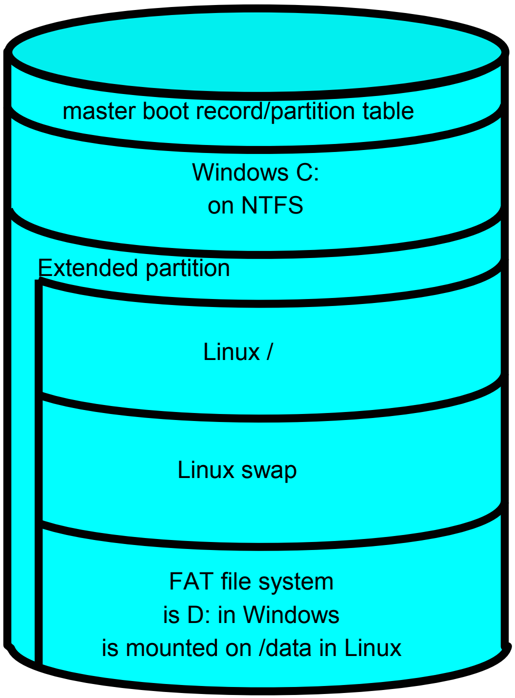
- `/opt` used for some software from external providers
  - Separate file system advisable
- Whatever you create for yourself

Typical file system layout

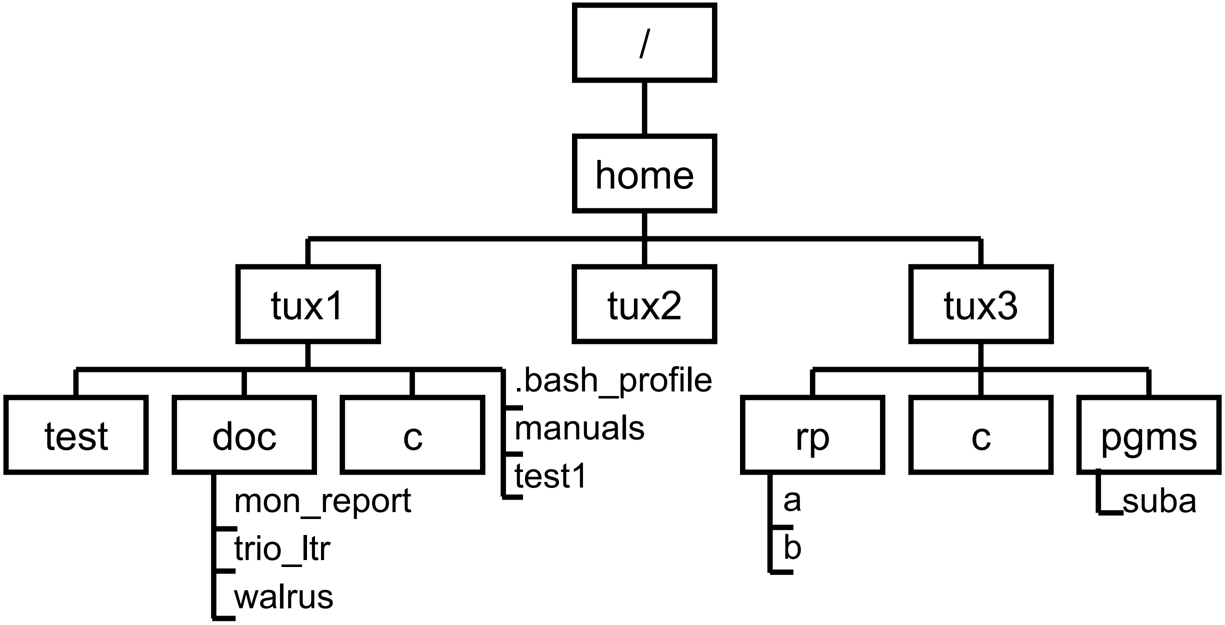
Typical Linux server



Dual-boot workstation



Example directory structure



Linux path names

Full path names:
Start from / (the root directory)

Relative path names:
Start from the present working directory

- Examples (working directory is `/home/tux1`):

– `/home/tux1/doc/mon_report`	(full)
– `doc/mon_report`	(relative)
– `../tux3/pgms/suba`	(relative)
– `./test`	(file in the current dir)
– `~/test`	(file under home directory)

Where am I?

- The **pwd** command (Print Working Directory) can be used to find out what your current working directory is.

```
$ pwd
/home/tux1
$
```

Change current directory

- With the **cd** (Change Directory) command:
  - `$ cd dir_name`

```
$ cd doc                        (relative)
$ cd /home/tux1/doc             (full)
$ cd ~tux1/doc                  (home)

$ cd     (Go to your home directory)
$ cd ..  (Go one directory up)
$ cd -   (Go to previous directory)
```

Create directories

- With the **mkdir** (Make Directory) command:
  - `$ mkdir dir_name`

```
$ mkdir /home/tux1/doc (full pathname)

$ cd /home/tux1
$ mkdir doc             (relative pathname)
```

Removing directories

- With the **rmdir** (Remove Directory) command:
  - `$ rmdir dir_name`

```
$ pwd
/home/tux1
$ rmdir doc test
rmdir: doc: Directory not empty
$

directory must be empty!
```

Working with multiple directories

- Create and remove multiple directories simultaneously with the -p flag.

```
$ mkdir -p dir1/dir2/dir3
$ rmdir -p dir1/dir2/dir3
```

List the contents of directories

- With the **ls** command:
  - `ls [ dir/file ]`

```
$ ls /home
tux1  tux2  tux3

Important options:
-l          long listing (more information)
-a          lists all files (including hidden)
-t          lists files sorted by change date
-R          lists contents recursively
```

An inode

- All files in Linux have the following attributes:
  - File type
  - Permissions (read, write, and so on)
  - Owner
  - Group
  - File size
  - File access, change/modification time
  - File deletion time
  - Number of links (soft/hard)
  - Extended attributes
  - Access control lists (ACLs)
- All of this information is stored in a system *index node* (inode). Each inode is identified by a unique number.
- An *inode* is a file's attributes and a pointer to the file data.

The touch command

- The **touch** command creates an empty file or updates the modification time of an existing file.

```
$ ls -l
-rw-rw-r-- 1 tux1 penguins  512 Jan 1 11:10 docs

$ touch docs
$ ls -l
-rw-rw-r-- 1 tux1 penguins  512 Jan 1 15:37 docs

$ touch new
$ ls -l
-rw-rw-r-- 1 tux1 penguins  512 Jan 1 15:37 docs
-rw-rw-r-- 1 tux1 penguins    0 Jan 1 15:38 new
```

Copying files (1 of 2)

- The **cp** command copies files.
  - `cp source[s]  [target]`

```
Copying one file to another:
$ cp  .bashrc  bashrc.old

Copying multiple files into a target
 directory:
$ cp  doc/mon_report  doc/walrus   /tmp
```

Copying files (2 of 2)

- **cp** can recursively copy directories with the -R flag.

```
$ cp -R /home/tux1/doc  /tmp

To prevent cp from overwriting existing
files, use:
$ cp -R -i /home/tux1/doc  /tmp
cp:  overwrite `/tmp/doc/walrus´?
```

Moving and renaming files (1 of 2)

- With the **mv** command:
  - `mv source[s] [target]`

```
To move a file do another directory:
$ mv doc/walrus  ../../tmp

To rename a file:
$ mv doc documents

Use the -i option to prevent mv from
overwriting existing files!
```

Moving and renaming files (2 of 2)

- Moving and renaming files can be combined by **mv**.

```
$ cd
$ pwd
/home/tux1
$ mv /tmp/walrus      ./test/walrus2

To move a directory:
$ mv ./test /tmp

mv is recursive by default
```

Removing files

- You can move files with the **rm** command.

```
$ rm test/walrus2
$ ls test/walrus2
ls: rob: No such file or directory

If unsure, use -i option
$ rm -i test/walrus2
rm: remove `test/walrus2´?

To remove files and directories recursively:
$ rm -ir test/
```

Listing file contents

- With the **cat** (Concatenate) command:

```
$ cat file1 file2 ...

$ cat walrus
"The time has come", the walrus said,
"To talk of many things:
Of shoes - and ships - and sealing wax -
Of cabbage - and kings -
And why the sea is boiling hot -
And whether pigs have wings."
$
```

Displaying files page by page

- With the **more** or **less** commands:

```
$ less walrus
"The time has come", the walrus said,
"To talk of many things:
Of shoes - and ships - and sealing wax -
Of cabbage - and kings -
And why the sea is boiling hot -
And whether pigs have wings."
/tmp/test/walrus 1-6/6 (END)
```

Displaying binary files

- With the **od** command:

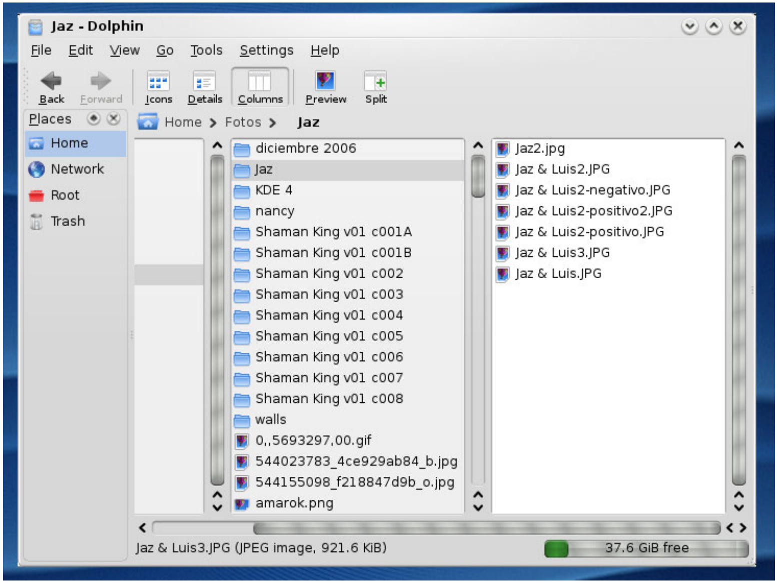
```
$ od /usr/bin/passwd
0000000 042577 043114 000401 000001 000000 000000 000000 000000
0000020 000002 000003 000001 000000 107300 004004 000064 000000
0000040 051430 000000 000000 000000 000064 000040 000006 000050
$
```

- With the **strings** command:

```
$ strings /usr/bin/passwd
/lib/ld.so.1
__gmon_start__
__deregister_frame_info
__register_frame_info
...
$
```

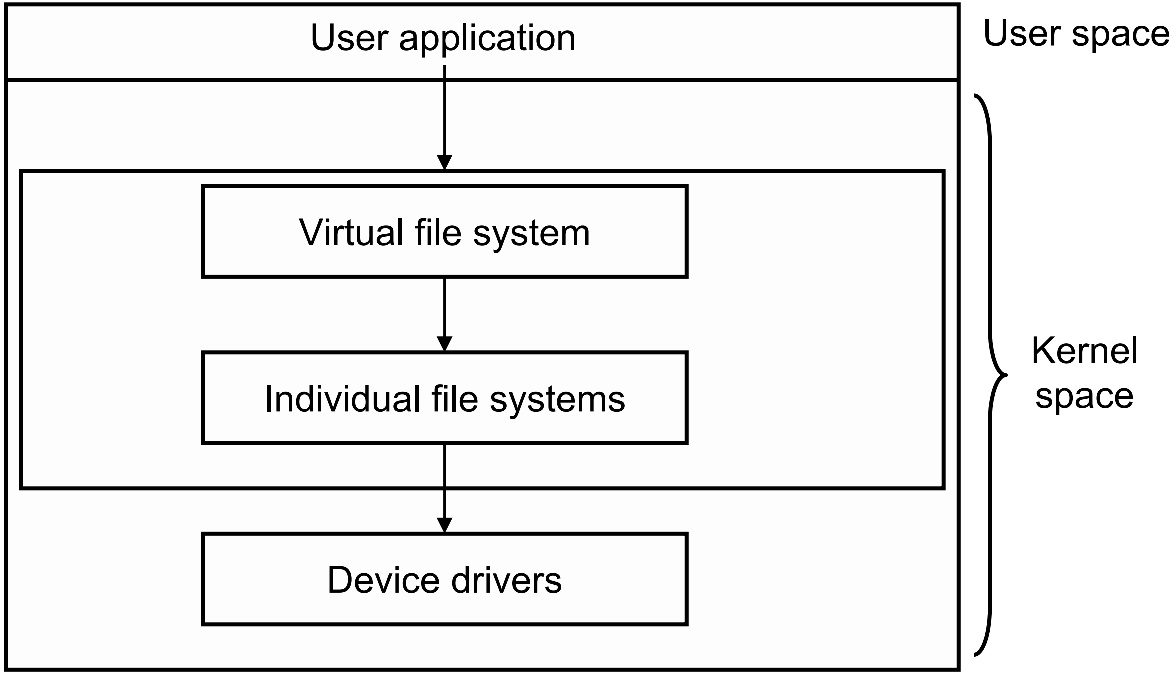
File managers

- Linux also offers different graphical file managers.
  - Nautilus (GNOME)
  - Konqueror (KDE)



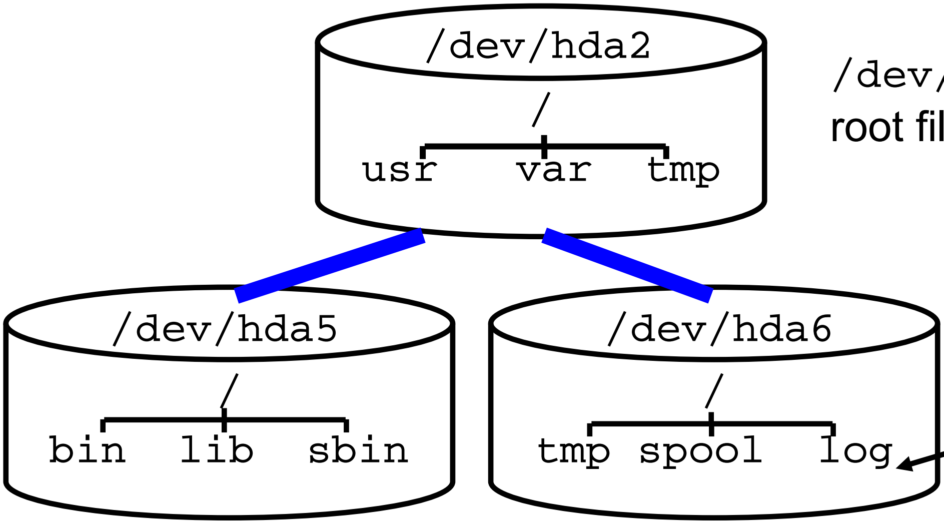
Virtual unified file system (1 of 2)

- Linux does not use drive letters (A:, C:, D:) to identify drives and partitions, but creates a virtual, unified file system.



Virtual unified file system (2 of 2)

- Different drivers and partitions are mounted on a mountpoint.
- Mounting associates a storage device with a file system.



`/dev/hda2` is the root file system.

This directory is now called `/var/log`.

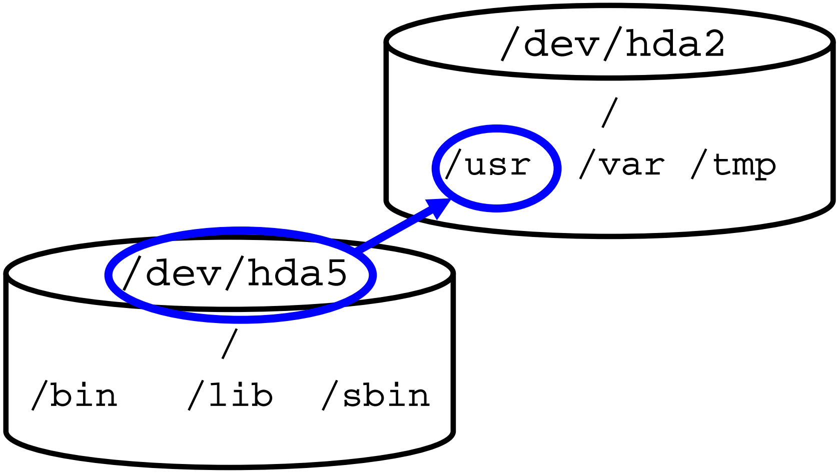
`/dev/hda5` is mounted on `/usr`.

`/dev/hda6` is mounted on `/var`.

The mount command

- The **mount** command mounts a file system.
  - Makes it part of the unified file system structure
  - `mount [-t type] [-o opts] device mountpnt`

```
# mount /dev/hda5 /usr
```



The umount command

- The **umount** command unmounts a file system.
  - It takes it out of the unified file system structure.
  - The file system should not be busy.
  - `umount {device|mountpnt}`

```
# umount /dev/hda5
- OR -
# umount /usr
```

The /etc/fstab file

- `/etc/fstab` lists all known file systems on the system.
- Syntax:
  – `device mountpoint type options dump fsck`
- File systems with the noauto option are not mounted automatically but can be used as templates for mount.

```
# cat        /etc/fstab
/dev/hda1    /mnt/winC       vfat     defaults           0 0
/dev/hda2    /               ext3     defaults           1 1
/dev/hda5    /usr            ext3     defaults           1 2
/dev/hda6    /var            ext3     defaults           1 2
/dev/cdrom   /media/cdrom    iso9660  noauto,owner,ro    0 0
/dev/fd0     /media/floppy   auto     noauto,owner       0 0
none         /proc           proc     defaults           0 0
none         /dev/pts        devpts   gid=5,mode=620     0 0
```

Note: Some distributions use file system labels instead of device names!

Mounting and unmounting removable media

- Most distributions configure `/etc/fstab` so that the console user is allowed to mount removable media (floppy, CD-ROM) on a predetermined mountpoint and with predetermined options (for security).
- Always unmount media before ejecting.
- The GUI typically mounts media automatically or nearly so.

```
$ whoami
tux1
$ mount /media/cdrom
$ mount
.
/dev/cdrom on /media/cdrom type iso9660 (ro,nosuid,nodev,user=tux1)
.
$ ls /media/cdrom
.

$ umount /media/cdrom
```

Hard links and soft (symbolic) links

- A *hard link* associates another file with an existing inode.
  - You cannot create a hard link for a directory or across file systems.
  - The file is not removed until all hard links to the file are removed.
- *Soft links* are like shortcuts to files or directories.
  - You can link to directories and across file systems.
  - They becomes useless when you remove the target file.

```
$ ln FileA FileB
$ ls -il FileA FileB
8986669 -rw-r—r--  2 test test 200 2010-04-22 15:15 FileA
8986669 -rw-r-r--  2 test test 200 2010-04-22 15:15 FileB

$ ln -s FileB FileC
$ ls -il FileB FileC
8986669 -rw-r-r-- 2 test test 200 2010-04-22 15:15 FileB
8986670 lrwxrwrwx 1 test test   5 2010-04-22 15:16 FileC -> FileB
```

Unit review

- There are three types of files.
  - Ordinary
  - Directory
  - Special
- The Linux file system structure is a hierarchical tree.
- Files are accessed using either full or relative path names. A full path name always begins with a forward slash (/).
- The following commands can be used with directories: **pwd**, **cd**, **mkdir**, **rmdir**, **touch**, and **ls**.
- The following commands can be used with files: **cat**, **more**, **less**, **cp**, **mv**, **rm**, **touch**, **od**, and **strings**.

Checkpoint

1. True or False: Linux imposes an internal structure on a regular file (not a directory or special file).

2. Which of the following is not a legal file name?
   a. `~tux1/mydocs.tar.gz`
   b. `/home/tux1/mydoc(1)`
   c. `/var/tmp/.secret.doc`
   d. `/home/../home/tux1/one+one`

3. What command would you use to copy the file `/home/tux1/mydoc` to `/tmp` and rename it at the same time to `tempdoc`?

Checkpoint solutions

1. True or False: Linux imposes an internal structure on a regular file (not a directory or special file).

   The answer is false.

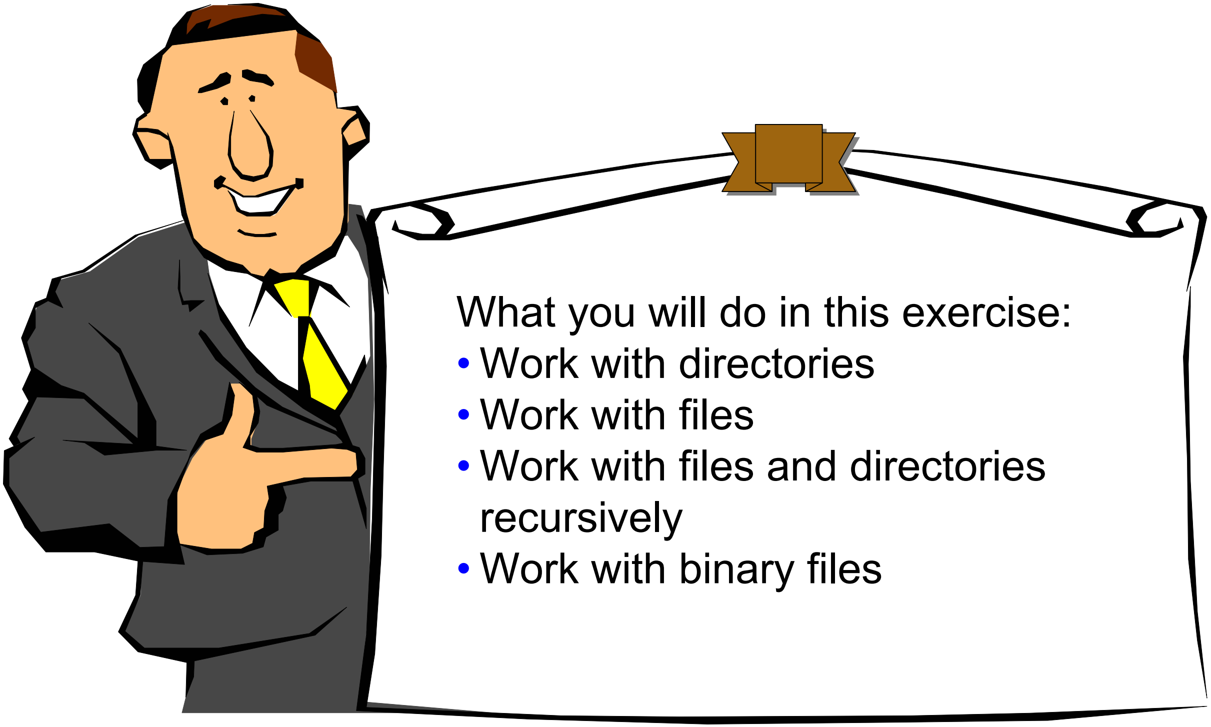
2. Which of the following is not a legal file name?
   a. `~tux1/mydocs.tar.gz`
   b. `/home/tux1/mydoc(1)`
   c. `/var/tmp/.secret.doc`
   d. `/home/../home/tux1/one+one`

   The answer is `/home/tux1/mydoc(1)`.

3. What command would you use to copy the file `/home/tux1/mydoc` to `/tmp` and rename it at the same time to `tempdoc`?

   The answer is `cp /home/tux1/mydoc/tmp/tempdoc`.

Exercise: Working with files and directories



What you will do in this exercise:

- Work with directories
- Work with files
- Work with files and directories recursively
- Work with binary files

Unit summary

Having completed this unit, you should be able to:

- Describe the different file types
- Describe file and path names
- Create, delete, copy, move, and list directories
- Create, delete, copy, and move files
- View the content of both text and binary files