Linux File System

by Mohamad H. Danesh

What is File System?

- It is responsible for storing information on disk and retrieving and updating this information.
- Example :
  - FAT16, FAT32, NTFS
  - ext2, ext3
- In Linux everything is file.

Type of File System

- Network File System
    - NFS
    - SMB
- Disk File System
    - ext2
    - ext3
    - FAT32
    - NTFS

Network File System

- Network File System are physically somewhere else, but appear as if they are mounted on one computer.
- NFS
  - It was developed by Sun.
- SMB
  - It was developed by Microsoft.

Disk File System

- Disk File System are what you will find on a physical device, such as hard drive in a computer.

ext2 File System

- It has been the standard File System for Linux.
- The original Extended File System was named ext.
- The ext2 File System can accommodate:
  - Files as large as 2GB
  - Directories as large as 2TB
  - Max. file name length of 255 characters.

Hard Disk Partitions

- Disk divided into partitions
  - Definition of partition: group of adjacent cylinders
  - 1 file system may reside in a single partition
- BIOS defines boot sector to be head 0, cylinder 0, sector 1
- Master Boot Record (MBR) – used to boot computer
- Partition table
  - Start and end addresses of each partition
  - One partition is marked as active
- When computer boots up:
  - BIOS executes MBR
  - MBR locates active partition and executes boot block

Hard Disk Partitions

- Each partition:
  - Starts with boot block even if it does not contain bootable OS
  - Superblock – key parameters about file system in partition such as magic # identifying file system type, number of blocks
  - Bitmap or linked-list of free blocks – free space management
  - Inodes – array of these (one per file)
  - Root directory of partition
  - Files and directories at the end

ext2 Structure

- A file in the ext2 File System begins with the inode.

- inode
  - Each file has an inode structure that is identified by an i-number.
  - The inode contains the information required to access the file.
  - It doesn’t contain file name.
  - Inodes store information on files, such as
    - user and group ownership,
    - access mode (read, write, execute permissions)
    - and type of file.

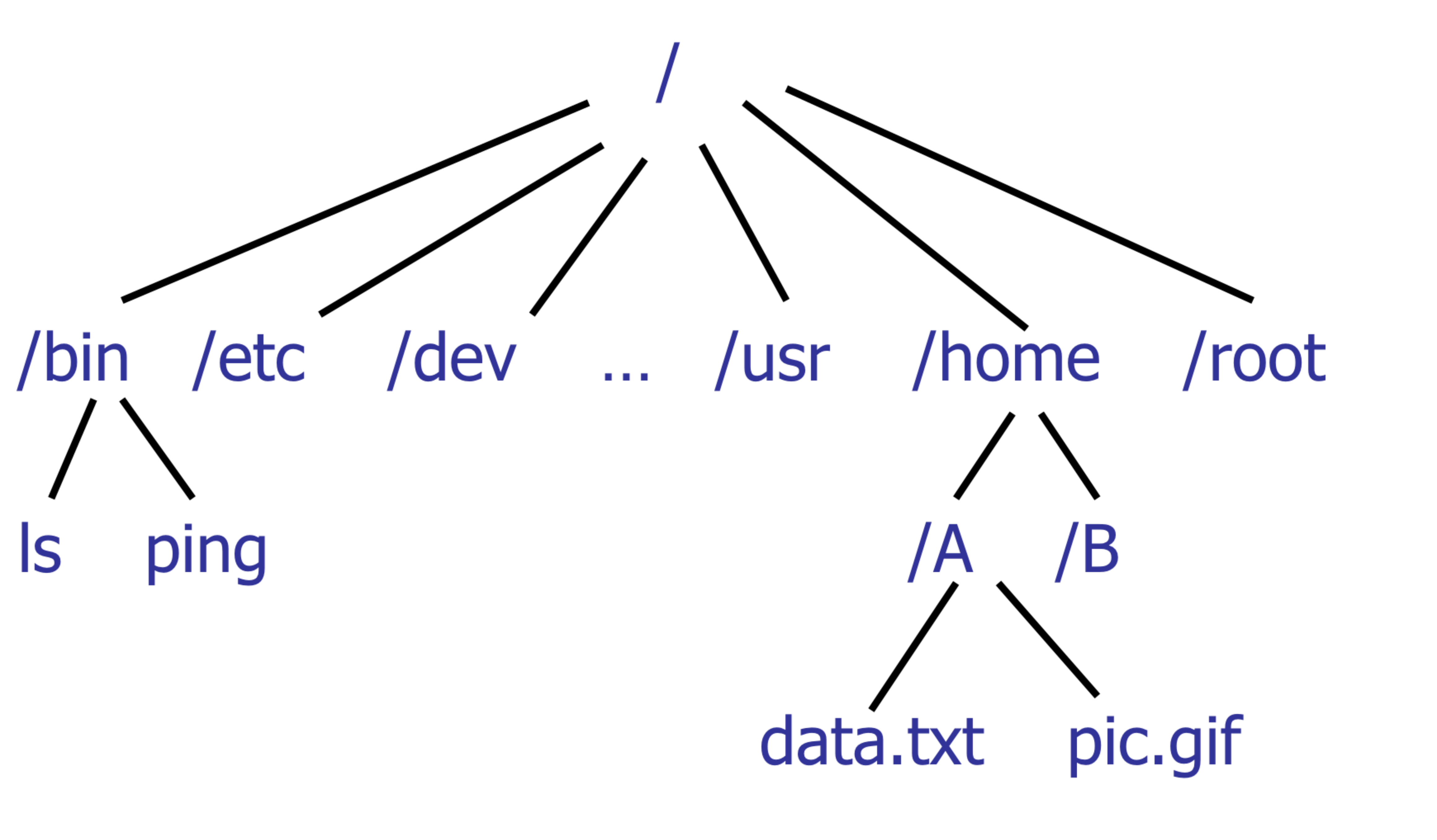
Physical Structure on the Disk

- Boot Block: information needs to boot the system
- Super Block: File System Specifications
  - Size
  - Max. number of files
  - Free blocks
  - Free inodes
- inode List
- Block List: The files data

ext3 File System

- It is as same as ext2.
- It is a journaling File System for Linux.
- In a journaling system, metadata is written to a journal on the disk before it is actually used to modify the file.

File System Structure



The Root Directory

- /bin
- /boot
- /dev
- /etc
- /home
- /initrd
- /lib
- /lost+found
- /media
- /mnt
- /opt
- /proc
- /root
- /sbin
- /usr
- /srv
- /tmp
- /var

/bin

- Hold the most commonly used essential user programs
  - login
  - Shells (bash, ksh, csh)
  - File manipulation utilities (cp, mv, rm, ln, tar)
  - Editors (ed, vi)
  - File system utilities (dd, df, mount, umount, sync)
  - System utilities (uname, hostname, arch)
  - GNU utilities like gzip and gunzip

/bin detail

- cat → Utility to concatenate files to standard output
- kill → Utility to send signals to processes
- chmod → Utility to change file access permissions
- chown → Utility to change file owner and group
- cp → Utility to copy files and directories
- ls → Utility to list directory contents
- mkdir → Utility to make directories
- mv → Utility to move/rename files
- pwd → Utility to print name of current working directory
- echo → Utility to display a line of text
- rm → Utility to remove files or directories

/sbin

- Hold essential maintenance or system programs such as the following:
  - fsck
  - Fdisk
  - Mkfs
  - Shutdown
  - Lilo
  - Init
- The main difference between the programs stored in /bin and /sbin is that the programs in /sbin are executable only by root.

/etc

- Store the systemwide configuration files required by many programs.
    - passwd
    - shadow
    - fstab
    - hosts
    - lilo.conf
    - …

/home and /root

- The /home directory is where all the home directories for all the users on a system are stored.
- The /root directory is where all the home directories for root user on a system are stored.

/dev

- The special files representing hardware are kept in it.
  - /dev/hda1
  - /dev/ttyS0
  - /dev/mouse
  - /dev/fd0
  - …

/tmp and /var

- The /tmp and /var directories are used to hold temporary files or files with constantly varying content.
- The /tmp directory is usually a dumping ground for files that only need to be used briefly and can afford to be deleted at any time.
- The /var directory is a bit more structured than /tmp and usually looks something like the following:
  - /var/log
  - /var/spool
  - /var/named
  - …

/usr

- Most programs and files directly relating to users of the system are stored.
- It is in some ways a mini version of the / directory.
  - /usr/bin
  - /usr/sbin
  - /usr/spool
  - …

/proc

- It is a virtual File System
- A special File System provided by the kernel as a way of providing information about the system to user programs.
- The main tasks of proc File System is to provide information about the kernel and processes.
- runtime system information (e.g. system memory, devices mounted, hardware configuration, etc).

Other directories

- /mnt
  - removable media such as CD-ROM, floppy and … are mounted.
  - /mnt/floppy
  - /mnt/cdrom
- /boot
  - Image to boot system
- /lost+found
  - Used by fsck

/lib

- Contains kernel modules and those shared library images (the C programming code library) needed to boot the system and run the commands in the root filesystem,
    - ie. by binaries in /bin and /sbin

- Windows equivalent to a shared library would be a DLL (dynamically linked library) file

Mounting File System

- The Linux File System makes it appear as if all the File System are local and mounted somewhere on the root File System.
- File System are mounted with the mount command.
  - mount –t type source mount_point
- To unmount a File System, the umount command is used.
  - umount /dev/<device name> or mount_point

Mounting Automatically with fstab

- This file lists all the partitions that need to be mounted at boot time and the directory where they need to be mounted.
- Along with that information, you can pass parameters to the mount command.
- /etc/fstab
  - Which devices to be mounted
  - What kinds of File Systems they contain
  - At what point in the File System the mount takes place
  - …

Partition Table

- MBR (Master Boot Record)
  - The first sector
  - 512 bytes (**446 bytes**:boot loader such as LILO or GRUB, **64 bytes**:partition table, **2 bytes**:special code).

- The partition table has enough room for four partitions.
  - One of the four can be used as an extended partition.

Partitions

- Primary-Master
  - /dev/hda
- Primary-Slave
  - /dev/hdb
- Secondary-Master
  - /dev/hdc
- Secondary-Slave
  - /dev/hdd
- Swap Partition
  - Used to implement virtual memory

Creating File System

- Once a disk has been partitioned for a specific File System, it is necessary to create a File System on it.
- The first process in the DOS world is known as formatting.
- In the UNIX world is known as creating a File System.

Create File System Commands

- mkfs or mke2fs
  - Make a new ext2 File System.
- mk3fs
  - Make a new ext3 File System.
- mkdosfs
  - Make DOS File System without owning any Microsoft software.

Formatting Linux Filesystem

- **Step #1 Create the new filesystem with following command (first login in as a root user)**
  - mkfs.ext3 /dev/sda5
- **Step # 2: Create mount point directory for the file system**
  - # mkdir /datadisk1
- **Step # 3: Mount the new file system**
  - # mount /dev/sda5 /datadisk1
- **Step # 4: Finally make sure file system /dev/hda5 automatically mounted at /datadisk1 mount point after system reboots.**
  - vi /etc/fstab
  - Add/append following entry to file:
  - /dev/sda5 /datadisk1 ext3 defaults 0 2

The shell

- The $ prompt that you see when you first log in is displayed by a shell (usually bash is Linux default shell).
- The Shell executes programs.
  - User types command
  - Shell reads command (read from input) and translates it to the operating system.
  - Shell types: Bash, csh, ksh, sh
- The `[username@server current_directory]$` signifies that
  - this console is being used by user `username`,
  - and the host-name is `server`.
  - The second word is the current working directory

Commands

- General command syntax format:

  $ command -options arguments

- Example:
  - `$ clear`
  - `$ cd /usr/src/linux`
  - `$ wc -w file1` (number of words in file1)
  - `$ wc -c file1` (number of characters in file1)
  - `$ wc -l file1` (number of lines in file1)
  - `$ cat file1 file2 file3`
  - `$ ls -al`

Wild-cards

```
$ ls -l *.c
$ ls [abc]*
$ ls ?a*
```

- * means 'match any number of characters'.
  - For example, chap* matches: *chap01, chapa, chap_end*, and also *chap*.
  - If you just give * (nothing else), it matches every file.
- ? means 'match any single character'.
  - For example, chap? matches: *chapa* and *chap1*, but not *chap01* and *chap*.

Wild-cards

- [..] means 'match any one characters between the brackets'. A range of characters may be specified by separating a pair of characters by a dash.
  - ➢For example, chap[abc] matches: *chapa* and *chapc*, but not *chap1* and *chapab*.
  - ➢[A-Za-z]* matches with any word whose first element is a character
  - ➢[!abc] matches with any one characters different from a,b and c

Control characters

- Interrupting
  - ➢ **^C** interrupts.
    - ✓ Exits the program and returns you to the command-line prompt.
  - ➢ **^Z** suspends.
    - ✓ Stops the program and puts it in the background. Type *fg* to restart it.
  - ➢ **^D** end of file.

- Stop/Start Scrolling: if what you want to see scrolls off the bottom of the screen, you may prevent this by sending a "stop" signal (**^S**) to the host. Send a "start signal to resume (**^Q**).
  - ➢ **^S** stops scrolling
  - ➢ **^Q** resume scrolling

Linux Help Manual

- The man command displays help manual for selected command:

man command_name

- `$ man man` or `$ man -help` help for the man command itself!!

Example: the command cat

```
[rinaldi@homelinux rinaldi]$ man cat


CAT(1)                    User Commands                    CAT(1)

NAME
       cat - concatenate files and print on the standard output

SYNOPSIS
       cat [OPTION] [FILE]...

DESCRIPTION
       Concatenate FILE(s), or standard input, to standard output.

       -A, --show-all
              equivalent to -vET

       -b, --number-nonblank
              number nonblank output lines

       -e     equivalent to -vE

       -E, --show-ends
              display $ at end of each line

       -n, --number
              number all output lines
```

```
 -s,       --squeeze-blank
      never more than one single blank line

-t     equivalent to -vT

-T,        --show-tabs
      display TAB characters as ^I

-u     (ignored)

-v,        --show-nonprinting
      use ^ and M- notation, except for LFD and TAB

--help display this help and exit

--version
      output version information and exit

With no FILE, or when FILE is -, read standard input.
```

```
AUTHOR
      Written by Torbjorn Granlund and Richard M. Stallman.

REPORTING BUGS
      Report bugs to <bug-textutils@gnu.org>.
```

COPYRIGHT

Copyright (C) 2002 Free Software Foundation, Inc.

This is free software; see the source for copying conditions. There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

SEE ALSO

The full documentation for cat is maintained as a Texinfo manual. If

the info and cat programs are properly installed at your site, the command *info cat* should give you access to the complete manual.

Options

- `$ man -f` *`command-name`* gives only a brief description of the command
  - man –f cat

- `$ whatis` *`command_name`* *(equivalent)*

  `man -f` and `whatis` display all the occurrences of the command in the chapters

- `$ man -k` *`keyword`* displays a list of commands in which description there is the word “keyword”
  - man -k cat
  - man -k manual

Linux Help Manual

- info is a program for reading the documentation about GNU utilities
  - ➢ `$ info info` (it shows an introduction to info)
  - ➢ `$ info emacs` (it describes how to edit with Emacs)
  - ➢ `$ info bash` (it gives a brief description of the Bash shell)
  - ➢ `$ info uname` (it prints information about the machine and the operating system it is run on)
    - ✓

```
$ uname -a
Linux homelinux 2.4.19-16mdk #1 Fri Sep 20
 18:15:05 CEST 2002 i686 unknown unknown GNU/
 Linux
```

Current Working Directory

- Every process has a location in the directory, termed its current working directory.
- When you log into a Linux system, your shell starts off in a particular directory called home directory (ex. /home/mohamad/).
- To display your current working directory use
  - ➢ `$ pwd`
- To come back the home directory use
  - ➢ `$ cd`
- Each file or directory has a unambiguously name specified by its pathname relative to / the root directory.
- A pathname relative the root directory is often termed an absolute pathname (ex. /usr/src/linux/CREDITS).
  - ➢ `$ cd /usr/src/linux/CREDITS`

Absolute and relative pathname

- A file can be unambiguously specified by using a pathname relative to its current working directory.
  - ➢ Example: if /usr/src/ is the current working directory, /linux/ CREDITS is the relative pathname of CREDITS

- The file system provides the following special fields that may be used supplying a relative path:
  - ➢ `.        - current directory`
  - ➢ `..       - parent directory`
  - ➢ `~        - your home directory`
  - ➢ Example:
    - ✓ `cd ..`
    - ✓ `cd ..`
    - ✓ `cd game`
    - ✓ `./fortune`

Listing the contents of a directory

- `$ ls` lists contents of current directory
- `$ ls dir_name` list `dir_name` contents
- ls options
  - -a all files including hidden files
    -C column list sorted down
    -F adds / for directory; * for executable; @ for symbolic links
    -l long format - file details
    -m across page; comma separators
    -padds / for directory; * for executable
    -r reverse alphabet order
    -R recursive; includes subdirectories
    -ssize of files in blocks
    -t list in time date last modified
    -u lists in time date last accessed
    -xcolumn list sorted across page
    -i inode of each file

Creating and Removing Directories

- `$ mkdir dir_name` (creates a directory)
  - `$ mkdir appunti ; ls`
  - `$ mkdir {appunti, lucidi};  ls`
- `$ rmdir dir_name` (removes empty directory. No warning!)
  - `$ rmdir appunti`
- If the directory is not empty, to remove it we must use `$ rm -r dir_name`
  - `$ rm -r appunti`
- It is not possible to remove the directories between home and /

Creating and Removing Files

- `$ cat > file_name` (store keyboard input into the file)
  - `$ cat > prova`

    ```
    Help! I'm stuck in a Linux program!

    ^D
    ```

    (The red text indicates what the user types.)

- `$ rm file_name` (removes file. No warning!)
  - `$ rm pippo` (Permanently removes file `pippo`. No warning!)
  - rm options:
    - -r (*recursive*): removes the contents of directories recursively
    - -i (*interactive*): prompts whether to remove each file
    - -f (*force*): forces rm to remove files independently from the permissions

Copy

- `$ cp options file1 file2` (to copy file1 into file2)
  ```
  $ cp /etc/passwd pass
  $ cp problemi/*   ~/backup
  ```
- If file2 does not exist, then cp creates it; otherwise cp overwrites it
- If file2 is a directory, cp makes a copy of file1 in the directory
  ```
  $ cp pippo ~/articoli
  $ cp /etc/passwd .
  ```
- cp options:
  - -i   prompt before overwriting existing file
  - -ppreserve permissions
  - -r   recursive copy files and subdirectories

Move

- `$ mv olddirectory newdirectory` (renames directory oldname to newname)
  - If `newdirectory` already exists mv moves `olddirectory` into the new one
- `$ mv oldname newname` (renames file oldname to newname)
  - If `newname` already exists mv writes `oldname` over `newname`
- mv options:
  - -i prompt before overwriting existing file
  - -f forces mv to replace reserve permissions
- `$ mv file path` (moves file in current directory to new directory)
  - `$ mv chap[1,3,7] book` (moves files chap1, chap3, and chap7 to directory book)
  - `$ mv chap[1-5] book` (moves files chap1 to chap5 to directory book)

Listing a file

- `$ cat filename` (displays the contents of filename)
- `$ more filename` (displays first screen of filename use space bar to scroll up one screen
  quits automatically after last screen)
- `$ less filename` (displays first screen of filename use space bar to scroll up one screen
  use up and down arrows to move up or down one line need to type q or Q to exit less command)
- `$ head -n filename` (displays the first n lines of filename. If n is not specified, it defaults to 10)
- `$ tail -n filename` (displays the last n line of filename. If n is not specified, it defaults to 10)

Standard Input and Standard Output

- Every program you run from the shell opens three files:
  - ➢ standard input ← 0
  - ➢ standard output ← 1
  - ➢ standard error ← 2
- The files provide the primary means of communications between the programs, and exist for as long as the process runs.
- The standard input file provides a way to send data to a process. As a default, the standard input is read from the *terminal keyboard*.
- The standard output provides a means for the program to output data. As a default, the standard output goes to the *terminal display screen*.
- The standard error is where the program reports any errors encountered during execution. By default, the standard error goes to the *terminal display.*

Redirecting Input and Output

- It is possible to tell a program:
  - where to look for input
  - where to send output,

  using input/output redirection. UNIX uses the special characters **<** and **>** to signify input and output redirection, respectively.
- Redirecting input: Using **<** with a file name (i.e., < file1) in a shell command tells the shell to read input from a file called "file1" instead of from the keyboard.

  ```
  $ more < /etc/passwd
  ```
- Redirecting output: Using **>** with a file name (i.e., > file 2) causes the shell to place the output from the command in a file called "file2" instead of on the screen. If the file "file2" already exists, the old version will be overwritten.

  ```
  $ ls /tmp > ~/ls.out
  $ sort pippo > pippo.ordinato
  ```

Redirecting Input and Output

- Use **>>** to append to an existing file (i.e., >> file2) causes the shell to append the output from a command to the end of a file called "file2".
  - ➢ If the file "file2" does not already exist, it will be created.

- Example

```
1. $ ls /bin  > ~/bin;  wc -l ~/bin
 $ ls /usr/sbin > ~/bin ; wc -l ~/bin

2. $ ls /bin  > ~/bin;
 $ ls /usr/sbin >> ~/bin; wc -l ~/bin
```

Redirecting Error

- Using **>&** with a file name (i.e., >& file1) causes the shell to place the standard error and the standard output from the command in a file called "file1".
  - If the file "file1" already exists, the old version will be overwritten.

- Example

```
$ ls abcdef
$ ls abcdef >& lserror
cat lserror
$ abcdef >& command
cat command
$ mkdir /bin/miei >& ~/miei; cat ~/miei
$ rm /bin/perl >& ~/errperl; cat ~/errperl
```

Pipes

- UNIX allows you to connect processes, by letting the standard output of one process feed into the standard input of another process. That mechanism is called a **pipe (|)**.

- `$ command1 | command2` causes the standard output of `command1` to flow through to standard input of `command2`.

- A sequence of commands chained together in this way is called a pipeline. Connecting simple processes in a pipeline allows you to perform complex tasks without writing complex programs.
  - ➢ `$ cat /etc/passwd | sort > ~/pass_ord`
  - ➢ `$ sort < pippo | lpr`

Exercises

1. Determine the number of files in the directory /bin whose first letter is “c”
2. Create a file containing the names of the first 7 files of the directory /etc
3. Determine the number of files of the directory /etc that contain “.”
4. Create a file containing a list with the name of 10 commands of /bin sorting by the last access time
5. Create a file containing the names of the first 7 files and the last 6 files (sorted in alphabetical order) of the directory /etc
6. Create a file containing a list with the name of 8 files in /usr/sbin/ sorting by the last modification time