Working with Files

Bok, JongSoon
javaexpert@nate.com
https://github.com/swacademy/Ubuntu

Understanding File Types

- From the command line, there are many ways you can create, find, and list different types of files.
  - Directories
  - Regular files
  - Device files
  - Hard links
  - Soft links
  - Named pipes
  - Sockets

Using Regular Files

- Regular files consist of data files
  - documents, music, images, archives, etc…
  - commands (binaries and scripts).
- Can determine the type of a file using the `file` command.

```
$ cd /usr/share/doc
$ file base-files/README
$ file base-files/copyright
$ file base-files/changelog.gz
$ file shared-mime-info/shared-mime-info-spec.html
$ file shared-mime-info/shared-mime-info-spec.pdf
```

Using Regular Files (Cont.)

- The `file` command that was run shows document files in the Ubuntu documentation directories of different formats.

```
$ touch /tmp/newfile.txt
$ > /tmp/newfile2.txt
$ ls -l /tmp/newfile2.txt
```

- A ***dash*** in the first character of the 10-character permission information (***-rw-r--r--***) indicates that the item is a regular file.

Using Regular Files (Cont.)

```
$ ls -l /usr/bin/apt-key

$ file /usr/bin/apt-key

$ file /bin/ls
```

`ls` Command

- List directory contents.
- Syntax
  - `ls [OPTION]... [FILE]...`
- Options
  - `-a,--all` : Do not hide entries starting with `.`.
  - `--author` : Print the author of each file.
  - `-d,--directory` : List directory entries instead of contents
  - `-F,--classify` : Append indicator (one of */=@|) to entries
  - `-h,--human-readable` : Print sizes in human readable format (e.g., 1K 234M 2G).

`ls` Command (Cont.)

- Options
  - **`-i,--inode`** : Print index number of each file.
  - **`-l`** : Use a long listing format.
  - **`-n, --numeric-uid-gid`** : like **`–i`**, but list numeric UIDs and GIDs.
  - **`-r,--reverse`** : Reverse order while sorting.
  - **`-R,--recursive`** : List subdirectories recursively.
  - **`-s,--size`** : Print size of each file, in blocks.
  - **`-S`** : Sort by file size.
  - **`-t`** : Sort by modification time.
  - **`-u`** : With **`-lt`**: sort by, and show, access time.
    with **`-l`**: show access time and sort by name
    otherwise: sort by access time

`ls` Command (Cont.)

- Options
  - **`-U`** : Do not sort ; list entries in directory order.
  - **`-X`** : sort alphabetically by entry extension.

`ls` Command (Cont.)

- Examples
  - `$ ls -lhRS /sbin`
  - `$ ls -l $HOME`
  - `$ ls -lr $HOME`
  - `$ ls -al $HOME`
  - `$ ls -lF $HOME`
  - `$ ls -il $HOME`
  - `$ ls -ln $HOME`
  - `$ ls -lR $HOME`
  - `$ ls -lSr $HOME`
  - `$ ls -lX /usr/share/doc/doc-base`
  - `$ la -clt $HOME`

`cat` Command

- Concatenate and write files.
- Syntax
  - `cat [OPTION][FILE]...`
- Options
  - `-b,--number-nonblank` : Number all nonempty output lines, starting with 1.
  - `-n,--number` : Number all output lines, starting with 1.

`cat` Command (Cont.)

- Examples
  - `$ cat > sample`
  - `$ cat sample`
  - `$ cat sample > sample1`
  - `$ cat test >> sample1`
  - `$ cat -b sample`
  - `$ cat text1 text2 text3 >> all_text`
  - `$ cat -n sample`

`more` Command

- Is a filter for paging through text one screenful at a time.
- Syntax
  - `more [OPTION] [+/pattern] [+linenum]`
- Options
  - **`-d`** : Will prompt the user with the message "[Press space to continue, 'q' to quit.]" and will display "[Press 'h' for instructions.]" instead of ringing the bell when an illegal key is pressed.
  - **`-n`** : Search for *k*th occurrence of last line. Defaults to 1.

`more` Command (Cont.)

- Commands
  - `[return]` : Display next k lines of text. Defaults to 1. Argument becomes new default.
  - `d` or `^D` : Scroll k lines. Default is current scroll size, initially 11. Argument becomes new default.
  - `[space]` : Display next k lines of text. Defaults to current screen size.
  - `b, ^B` : Skip backwards k screenfuls of text. Defaults to 1. Only works with files, not pipes.
  - `h` : Help: display a summary of these commands. If you forget all the other commands, remember this one.

`more` Command (Cont.)

- Commands
  - `v` : Start up an editor at current line. The editor is taken from the environment variable VISUAL if defined, or EDITOR if VISUAL is not defined, or defaults to "vi" if neither VISUAL nor EDITOR is defined.
  - `:f` : Display current file name and line number.
  - `=` : Display current line number.
  - `Q,q` : Exit.
  - `/  [pattern]` : Search specified word.

`more` Command (Cont.)

- Examples
  - `$ more /etc/services`
  - `$ more -d +10 /etc/services`
  - `$ more -5 /etc/services`

`head` Command

- Prints the first part (10 lines by default) of each FILE.
- If more than one FILE is specified, prints a one-line header consisting of :

  ```
  ==> FILE NAME <==
  ```

  before the output for each FILE.
- Syntax
  - `head [OPTION]... [FILE]`
- Options
  - `-n` : Output the first n lines.
  - `-c` : Print the first K bytes of each file.

`head` Command (Cont.)

- Examples
  - `$ head /etc/passwd`
  - `$ head -3 /etc/passwd`
  - `$ head /etc/*.conf`
  - `$ head -3 /etc/*.conf`
  - `$ head /etc/services`
  - `$ head –c 200 /etc/services`

`tail` Command

- Prints the last part (10 lines by default) of each FILE.
- If more than one FILE is specified, prints a one-line header consisting of :

  ==> FILE NAME <==

  before the output for each FILE.
- Syntax
  - **`tail [OPTION]... [FILE]`**
- Options
  - **`-n`** : Output the first n lines.
  - **`-f`** : Output appended data as the file grows.
  - **`-c`** : Output the last N bytes.

`tail` Command (Cont.)

- Examples
  - `$ tail /etc/passwd`
  - `$ tail -3 /etc/passwd`
  - `$ tail /etc/*.conf`
  - `$ tail -f /etc/*.conf`

Using Directories

- Is a container for files and subdirectories.
- Are set up in a hierarchy from the root (`/`) down to multiple subdirectories, each separated by a slash (`/`).
- Are called ***folders*** when you access them from graphical file managers.
- To create new directories for storing your data, you can use the **`mkdir`** command.

```
$ mkdir /tmp/new
$ mkdir -p /tmp/a/b/c/new
$ mkdir -m 700 /tmp/new2
```

`mkdir` Command

- Make directory(ies).
- Syntax
  - `mkdir [OPTION] DIRECTORY`
- Options
  - `-m,--mode=MODE` : Set permission mode (as in chmod), not `rwxrwxrwx` - umask.
  - `-p,--parents` : No error if existing, make parent directories as needed.

`cd` Command

- Changes working directory.
- Absolute Path versus Relative Path.
- Syntax
  - `cd [directory]`

`cd` Command (Cont.)

- Examples
  - `$ cd`
  - `$ cd $HOME`
  - `$ cd ~`
  - `$ cd ~chris`
  - `$ cd -`
  - `$ cd $OLDPWD`
  - `$ cd ..`
  - `$ cd /usr/bin`
  - `$ cd usr/bin`

`rmdir` Command

- Remove *empty* directory(ies).
- Syntax
  - `rmdir [OPTION] DIRECTORY`
- Options
  - `-m,--mode=MODE` : Set permission mode (as in chmod), not `rwxrwxrwx` - umask.
  - `-p,--parents` : Remove DIRECTORY, then try to remove each directory component of that path name.

`rmdir` Command (Cont.)

- Examples
  - `$ mkdir -p a/b/c/new`
  - `$ touch a/b/c/new/test.txt`
  - `$ rmdir a`
  - `$ rmdir -p a/b/c/new`
  - `$ rm a/b/c/new/test.txt`
  - `$ rmdir -p a/b/c/new`

`cp` Command

- Copy files and directories.
- Syntax
  - `cp [OPTION] SOURCE DESTINATION`
- Options
  - `-a,--archive` : Preserve as much as possible of the structure and attribute of the original files.
  - `-b` : Make a backup of each file that would otherwise be overwritten or removed.
  - `-f,--force` : If an existing destination file cannot be opened, remove it and try again.
  - `-i,--interactive` : Prompt before overwrite.
  - `-r,--recursive` : Copy directories recursively.

`cp` Command (Cont.)

- Examples
  - `$ mkdir -p a/b/c/new`
  - `$ touch a/b/c/new/test.txt`
  - `$ cp a/b/c/new/test.txt .`
  - `$ cp a copy_a`
  - `$ cp -r a copy_a`
  - `$ cp -a file file_copy`
  - `$ cp -b file file_copy`

`rm` Command

- Remove files and directories.
- Syntax
  - `rm [OPTION] FILE`
- Options
  - `-i,--interactive` : Prompt before any removal.
  - `-f,--force` : Ignore nonexistent files, never prompt.
  - `-r,--recursive` : Remove the contents of directories recursively.
- Examples
  - `$ rm -rf sample`
  - `$ rm -i imsi*`

`mv` Command

- Rename SOURCE to DEST, or move SOURCE(s) to DIRECTORY.
- Syntax
  - `mv [OPTION] SOURCE DEST`
  - `Mv [OPTION] SOURCE DIRECTORY`
- Options
  - `-i,--interactive` : Prompt before overwriting.
  - `-f,--force` : Do not prompt before overwriting.
- Examples
  - `$ mv sample  sample1`
  - `$ mv sample $HOME`

`chmod` Command

- Change file access permission.
- Syntax
  - `chmod [OPTION] MODE[,MODE] FILE`
  - `chmod [OPTION] OCTAL-MODE FILE`
- Options
  - `-R,--recursive` : Change files and directories recursively.

`chmod` Command (Cont.)

- Symbolic Mode.
- Syntax
  - `chmod [ugao][+-=][rwx] FILE`

Option	Description	Option	Description	Option	Description
u	User	+	Add	r	Read
g	Group	-	Remove	w	Write
o	Other User	=	Cancel	x	eXecute
a	All				

`chmod` Command (Cont.)

- Octal Mode.
- Syntax
  - `chmod [octal-number] FILE`

Option	Description
7	`rwx`
6	`rw`
5	`r-x`
4	`r--`
3	`-wx`
2	`-w-`
1	`--x`
0	`---`

`chmod` Command (Cont.)

- Examples
  - `$ chmod 700 sample`
  - `$ chmod 711 sample`
  - `$ chmod go+r sample`
  - `$ chmod 777 sample`
  - `$ chmod a=rwx sample`
  - `$ chmod 000 sample`
  - `$ chmod a-rwx sample`
  - `$ chmod go-rw sample`

`ln` Command

- Make links between files.
- Syntax
  - `ln [OPTION] SOURCE DEST`
- Options
  - `-s,--symbolic` : Make symbolic links instead of hard links.

`ln` Command (Cont.)

- Soft link (Symbolic link)
  - Points to a file or change to one that points to a directory.
  - The target has its own set of permissions and ownership that you cannot see from the symbolic link.
  - Can exist on a different disk partition than the target.
  - In fact, the symbolic link can exist, even if the target doesn’t.

`ln` Command (Cont.)

- Hard link
  - Can only be used on files (not directories).
  - Is basically a way of giving multiple names to the same physical file.
  - Every physical file has at least one hard link, which is commonly thought of as the file itself.
  - Must be on the same partition as the original target file.
  - All have the same i-node number.
  - Changing permissions, ownership, date/time stamps or content of any hard link to a file results in all others being changed as well.

`ln` Command (Cont.)

- Samples
  - `$ cat > sample`
  - `$ ln sample  test`
  - `$ ls -il`
  - `$ vi test`
  - `$ cat sample`

  - `$ ln -s sample test`
  - `$ vi test`
  - `$ ls -il`
  - `$ rm sample`
  - `$ cat test`

`wc` Command

- Print the number of bytes, words, and lines in files.
- Syntax
  - `wc [OPTION] ... [FILE] ...`
- Options
  - `-c, --bytes` : Print the byte counts.
  - `-m, --chars` : Print the character counts.
  - `-l, --lines` : Print the newline counts.
  - `-w, --words` : Print the word counts.

`wc` Command (Cont.)

- Examples
  - `$ ls -l /usr/share/doc/ufw/README.Debian`
  - `$ cd /usr/share/doc/ufw`
  - `$ wc -c README.Debian`
  - `$ wc -m README.Debian`
  - `$ wc -l README.Debian`
  - `$ wc -w README.Debian`
  - `$ wc copyright README.Debian`
  - `$ wc -c copyright README.Debian`

`sort` Command

- Sort lines of text files.
- Syntax
  - `sort [OPTION] … [FILE] …`
- Options
  - `-k, --key=POS1[,POS2]` : Start a key at POS1, end it at POS 2 (origin 1)
  - `-r, --reverse` : Reverse the result of comparisons.
  - `-u, --unique` : with `–c` : check for strict ordering

    otherwise : output only the first of an equal run

`sort` Command (Cont.)

- Examples
  - `$ cat textfile`
  - `$ sort textfile`
  - `$ sort -r textfile`
  - `$ sort -k 2 textfile`
  - `$ sort -rk 2 textfile`
  - `$ sort -u textfile2`
  - `$ sort -ur textfile2`
  - `$ sort -uk 2 textfile2`
  - `$ sort /etc/passwd | head`
  - `$ ls -l /var/log | sort -k 5`

`cmp` Command

- Compare two files byte by byte.
- Syntax
  - `cmp [OPTION] file1 file2`
- Options
  - `-i` : Skip first bytes of both inputs.
  - `-l` : Print the byte number (decimal) and the differing byte values (octal) for each difference.
  - `-s` : Print nothing for differing files; return exit status only

    `0` : The files are identical.

    `1` : The files are different.

    `>1` : An error occurred.

`cmp` Command (Cont.)

- Examples
  - `$ cat file1`

    I am a LINUX System Engineer.
  - `$ cat file2`

    I am a Linux System Engineer.
  - `$ cmp file1 file2`
  - `$ cp file1 file3`
  - `$ cmp file1 file3`
  - `$ cmp -l file1 file2`
  - `$ cmp -i 1 file1 file2`
  - `$ cmp -i 8 file1 file2`

`diff` Command

- Compare files line by line.
- Syntax
  - `diff [OPTION] file1 file2`
- Options
  - `-q, --brief` : Report only when files differ.
  - `-c` : Use the context output format.
  - `-d` : Try hard to find a smaller set of changes.
  - `-r` : When comparing directories, recursively compare any subdirectories found.
  - `-s` : Report when two files are the same.

`diff` Command (Cont.)

- Examples
  - `$ cat file1`

```
Linux is a Operating System.

I am a Linux Engineer.
```

  - `$ cat file2`

```
Linux is a Operating System.

I am a Linux System Engineer.
```

  - `$ diff file1 file2`
  - `$ diff --brief file1 file2`
  - `$ diff -c file1 file2`
  - `$ diff -d file1 file2`

`diff3` Command

- Compare three files line by line.
- Syntax
  - `diff3 file1 file2 file3`

`diff3` Command (Cont.)

- Examples
  - `$ cat file1`

```
Linux is a Operating System.

I am a Linux Engineer.
```

  - `$ cat file2`

```
Linux is a Operating System.

I am a Linux System Engineer.
```

  - `$ cat file3`

```
Linux is a Operating System.

I am a Linux Engineer.
```

  - `$ diff3 file1 file2 file3`

Using Device Files

- When applications need to communicate with computer’s hardware, direct data to device files.
- Are stored in the `/dev` directory.
- Devices are generally divided into block devices (such as storage media) and character devices (such as serial ports and terminal devices).
- Are often called device drivers.
- In Linux and Unix, the operating system treats almost everything as a file, hence the term device files.

Using Device Files (Cont.)

- Is associated with :
  - A major number (indicating the type of device)
  - A minor number (indicating the instance number of the device).
- Terminal(tty) devices are represented by major character device 4.
- SCSI hard disks are represented by major block device number 8.

```
$ ls -l /dev/tty0 /dev/sda1
```

Using Named Pipes and Sockets

- When you want to allow one process to send information to another process, can simply pipe (|) the output from one to the input of the other.
- Named pipes are typically used for interprocess communication on the local system.
- Sockets can be used for processes to communicate over a network.

Using Named Pipes and Sockets (Cont.)

- Are often set up by applications in the `/tmp` directory.

```
$ ls -l /tmp/.X11-unix/X0
```

- To create your own named pipe, use the `mkfifo` command.

```
$ mkfifo mypipe
$ ls -l mypipe
```