Linux/Unix System Programming

CSCI 2153

David L. Sylvester, Sr., Professor

Files Management

- When working with Unix, you spend most of your time working with files. In Unix, there are three basic types of files –
- **Ordinary Files** – An ordinary file is a file on the system that contains data, text, or program instructions. In this tutorial, you look at working with ordinary files.
- **Directories** – Directories store both special and ordinary files. For users familiar with Windows or Mac OS, Unix directories are equivalent to folders.
- **Special Files** – Some special files provide access to hardware such as hard drives, CD-ROM drives, modems, and Ethernet adapters. Other special files are similar to aliases or shortcuts and enable you to access a single file using different names.

Files Management (cont.)

Listing Files

To list the files and directories stored in the current directory, use the following command −

```
$ls
```

Here is the sample output of the above command −

```
$ls

bin             hosts     lib       res.03
ch07            hw1       pub       test_results
ch07.bak        hw2       res.01    users
docs            hw3       res.02    work
```

Files Management (cont.)

The command **ls** supports the **-l** option which would help you to get more information about the listed files −

```
$ls -l
total 1962188
drwxrwxr-x  2  amrood  amrood    4096  Dec 25 09:59  uml
-rw-rw-r--  1  amrood  amrood    5341  Dec 25 08:38  uml.jpg
drwxr-xr-x  2  amrood  amrood    4096  Feb 15  2006  univ
drwxr-xr-x  2  root    root      4096  Dec  9  2007  urlspedia
-rw-r--r--  1  root    root    276480  Dec  9  2007  urlspedia.tar
drwxr-xr-x  8  root    root      4096  Nov 25  2007  usr
drwxr-xr-x  2  200     300       4096  Nov 25  2007  webthumb-1.01
-rwxr-xr-x  1  root    root      3192  Nov 25  2007  webthumb.php
-rw-rw-r--  1  amrood  amrood    20480  Nov 25  2007  webthumb.tar
-rw-rw-r--  1  amrood  amrood    5654  Aug  9  2007  yourfile.mid
-rw-rw-r--  1  amrood  amrood   166255  Aug  9  2007  yourfile.swf
drwxr-xr-x  11  amrood  amrood    4096  May 29  2007  zlib-1.2.3
$
```

Files Management (cont.)

Column Descriptions: **drwxrwxr-x 2 amrood amrood 4096 Dec 25 09:59 uml**

First Column − Represents the file type and the permission given on the file. Below is the description of all type of files.

Second Column − Represents the number of memory blocks taken by the file or directory.

Third Column − Represents the owner of the file. This is the Unix user who created this file.

Fourth Column − Represents the group of the owner. Every Unix user will have an associated group.

Fifth Column − Represents the file size in bytes.

Sixth Column − Represents the date and the time when this file was created or modified for the last time.

Seventh Column − Represents the file or the directory name.

In the **ls -l** listing example, every file line begins with a d, -, or l. These characters indicate the type of the file that's listed.

Files Management (cont.)

List of **ls** options:

-	Regular file, such as an ASCII text file, binary executable, or hard link.
B	Block special file. Block input/output device file such as a physical hard drive.
c	Character special file. Raw input/output device file; physical hard drive.
d	Directory file that contains a listing of other files and directories.
l	Symbolic link file. Links on any regular file.
p	Named pipe. A mechanism for interprocess communications.
s	Socket used for interprocess communication.

The * (asterisk) and **?** (question mark) are metacharacters. The * is used to match 0 or more characters, a question mark (**?**) matches with a single character.

$ls ch*.doc	Lists all files beginning with **ch** that has the extension **.doc**
$ls ch?.doc	List all files beginning with **ch** followed by one character that has the extension **.doc**

Files Management (cont.)

Hidden Files

An invisible file is one, the first character of which is the dot or the period character (.). Unix programs (including the shell) use most of these files to store configuration information.

Some common examples of the hidden files include the files −

- **.profile** − The Bourne shell (sh) initialization script
- **.kshrc** − The Korn shell (ksh) initialization script
- **.cshrc** − The C shell (csh) initialization script
- **.rhosts** − The remote shell configuration file

To list the invisible files, specify the **-a** option to **ls** −

$ ls –a

Single dot (.) − This represents the current directory.
Double dot (..) − This represents the parent directory.

Files Management (cont.)

Creating Files

The **vi** editor can be used to create ordinary files on any Unix system.
In the terminal type **vi filename** and press ENTER.
The **vi** command will open a file with the given filename. Now, press the key **i** to come into the edit mode. Once in the edit mode, start writing your content in the file as in the following program −

This is a unix file.... I created it for the first time..... I'm going to save this content in this file.

Once done typing, follow these steps −
Press the **ESC** key to come out of the edit mode.
Then while holding the **SHIFT** key, press **ZZ**, (Shift ZZ) to come out of the file completely.

Files Management (cont.)

Editing Files (using the vi editor**)**

Type:

$ **vi filename**

Then press the key **i** to start editing the file. To move the cursor throughout the file press the **ESC** key then use the following keys.

- **l** key to move to the right side.
- **h** key to move to the left side.
- **K** key to move upside in the file.
- **j** key to move downside in the file.

The above keys, allows you position your cursor wherever you want to edit. Once you positioned, use the **i** key to enter the edit mode. Once done with the editing, press the **Esc** key then, **Shift + ZZ** to save and exit the editor.

Register with CoCalc

- **Cocalc website: https://cocalc.com**
- **Register with CoCalc**
- **Create a CoCalc Project**
- **Create a terminal session in CoCalc**
- **Create a file using vi in CoCalc**
- **Creating a folder in CoCalc**
- **Executing a c++ proram in CoCalc**

- **Announce Assignment (vi Editor/ c++ Compiler)**

Files Management (cont.)

Counting Words in a File

The **wc** command displays a count of lines, words, and characters contained in a file.

```
$ wc filename
  2    19    103      filename
```

First Column – total number of lines in the file.
Second Column – total number of words in the file.
Third Column – total number of bytes in the file (file size).
Fourth Column – Represents the file name.

Multiple files can be listed to get information about those files at a time.

```
$ wc filename1 filename2 filename3
```

Files Management (cont.)

Copying Files

The **cp** command is used to copy files.

$ **cp source_file destination_file**

$ **cp filename copyfile** This command copies the content of the file named **filename** into a new file named **copyfile** into your current directory. This file will be an exact duplicate of the file **filename**.

Renaming Files

The **mv** command renames a file.

$ **mv old_file new_file** This command rename the existing file filename to newfile.

file **filename** to **newfile**.

$ **mv filename newfile** In the command the contents of **filename** (the existing file) is completely placed into the new file named **newfile** (yielding only newfile) in your current directory.

Files Management (cont.)

Deleting Files

The **rm** command is used to delete (remove) files.

$ **rm filename**

It may be better to use the **-i** option along with **rm** command. This option will cause the system to prompt you before removing (deleting) the file.

To completely remove the existing file **filename**, use the following command:

$ **rm filename**

Multiple files can be removed at one time using the following command:

$ **rm filename1 filename2 filename3**

Files Management (cont.)

Concatenate Files

cat stands for concatenation (to merge things together) and is one of the most useful and versatile Linux commands. The cat command can be used to support a number of operations utilizing strings, files, and output.

$ **cat filename** or $ **cat filename filename**

The cat command displays the content of one or more text files on the screen without pausing.

To create a file using the cat command, simply type

$ **cat filename > newfile**

The > symbol directs input to the file. If the filename already exists, the contents of the file will be overwritten.

$ **cat filename1 > filename2** (overwrites the content of filename2 with filename1)

Files Management (cont.)

$ **cat filename1 > filename2**

This command overwrites the contents of filename2 with filename1.

To append the contents of a file you would use the **>>** along with the **cat** command.

$ **cat filename1 >> filename2**

This command append at the bottom of filename2 the contents of filename1

Files Management (cont.)

more Command

$ **more filename**

The more command displays the content of a text file one screen at a time. The –More—(n%) message appears at the bottom of each screen, where **n%** is the percentage of the file that has been displayed. When the entire file has been displayed, the shell prompt appears.

When the –More–(n%) prompt appears at the bottom of the screen, you can use the keys described in the table on the following page to scroll through the file.

Files Management (cont.)

Keyboard Command	Action
Space bar	Moves forward one screen
Return	Scrolls one line at a time
b	Moves back one screen
h	Displays a help menu of features
/string	Searches forward for patter
N	Finds the next occurrence of pattern
Q	Quits and returns to the shell prompt

Files Management (cont.)

grep Command

$ **grep word filename**

The grep command, which means **global regular expression print**, remains amongst the most versatile commands in a Linux terminal environment. It happens to be an immensely powerful program that lends users the ability to sort input based on complex rules, thus rendering it a fairly popular link across numerous command chains. The grep command is primarily used to search text or search any given file for lines containing a match to the supplied words/strings. By default, grep displays the matching lines, and it may be used to search for lines of text matching one/many regular expressions in a fuss-free, and it outputs only the matching lines.

Files Management (cont.)

grep Command

$ **grep mike /etc/passwd**

This command search for the user “mike” in the Linux passwd file.

To instruct grep to ignore word case, i.e., match abc, Abc, ABC, and all possible combinations use the -i option.

$ **grep -i print sample.cpp**

This command search for the string “print” regardless of the letter case.

Files Management (cont.)

grep Command (Other options

- -w (search for a word)
- -c (count lines for matched words)
  -n (give line number)
- -v (inverted match)

Files Management (cont.)

Standard Unix Streams

Under normal circumstances, every Unix program has three streams (files) opened for it when it starts up.

- **stdin** − This is referred to as the *standard input* and the associated file descriptor is 0. This is also represented as STDIN. The Unix program will read the default input from STDIN.
- **stdout** − This is referred to as the *standard output* and the associated file descriptor is 1. This is also represented as STDOUT. The Unix program will write the default output at STDOUT
- **stderr** − This is referred to as the *standard error* and the associated file descriptor is 2. This is also represented as STDERR. The Unix program will write all the error messages at STDERR.