Linux Operating System and Programming

TOPIC# 4
The File System

The File

The file is a container for storing information (sequence of characters). A UNIX file doesn't contain the eof (end-of-file) mark. All the file attributes are kept in a separate area of the hard disk, not directly accessible to humans, but only to the kernel.

A directory is simply a folder where you store filenames and other directories. All physical devices like the hard disk, memory, CD-ROM, printer and modem are treated as files. Even shell and kernel are also files.

There are three types of files.

1. Ordinary file

This is also known as regular file. It contains only data as a stream of characters. An ordinary file is divided into two types:

a. Text file

A text file contains only printable characters. All C and java program sources, shell and perl scripts are text files. A text file contains lines of characters where every line is terminated with the *newline* character, also known as *linefeed*(LF). Normally this linefeed is not visible when you press enter key, but with **od** command it can be made visible.

b. Binary file

A binary file contains both printable and unprintable characters that cover the entire ASCII range (0 to 255). Most UNIX commands are binary files and the object code and the executables that are produced by compiling the C programs are also binary files. Picture, sound and video files are also binary files. If you try to display these files by simple **cat** command, it will produce unreadable output.

2. Directory file

A directory contains no data, but keeps some details of the files and subdirectories that it contains. A directory contains names of files and other directories and a number associated with each name. A directory file contains an entry for every file and subdirectory. Each entry has two components.

a. The filename
b. A unique identification number for the file or directory(called the inode number)

You cannot write a directory file, but you can perform some action that makes the kernel write a directory. For example, when a file is created or removed, kernel automatically updates its corresponding directory by adding or removing the entry (inode number and filename) associated with the file.

3. Device file

All devices and peripherals are represented by files. To read or write a device, one has to perform these operations on its associated file.

Device filenames are generally found inside a single directory structure, **/dev**. The operation of a device is entirely governed by the attributes of its associated file. The kernel identifies a device from its attributes and then uses them to operate the device.

4. Named pipe / fifo (First in first out)

The mkfifo program takes one or more file names as arguments for this task and creates pipes with those names. For example, to create a named pipe with the name testpipe give the command:
mkfifo testpipe

The simplest way to show how named pipes work is with an example. Suppose we've created testpipe as shown above. In one virtual console1, type:

ls -l > testpipe

and in another type:
cat < testpipe

The output of the command run on the first console shows up on the second console. Note that the order in which you run the commands doesn't matter.
Courtesy: http://www.linuxjournal.com/article/2156

ls -ltr testpipe
prw-rw-r-- 1 jpandya jpandya 0 Aug 8 12:47 testpipe

'p' indicates that its a fifo/named pipe type of file.
Real time application of fifo. The requirement was to migrate data from oracle database to vertica database. We wanted to have multiple threads run in parallel to complete the process fast and utilize all resources. There were 15 writer threads processes to read different blocks from oracle database table and write it to fifo (named pipe) and than other readers thread to read from fifo and write it to vertica database. Using this fifo, it was really awesome performance wise.

A filename:

A filename in UNIX can be up to 255 characters. A filename can contain practically any ASCII characters except the / and the NULL character. It can have as many as dots embedded in its name. UNIX is case sensitive, file01, File01 and FILE01 are three different filenames and they can co-exist in the same directory.

Never use a – at the beginning of a filename. A command that uses a filename as argument often treats it as an option and reports errors.

All files in UNIX are related to one another. The file system in UNIX is a collection of all of these related files (ordinary, directory and device files) organized in a hierarchical structure.

The HOME variable: The HOME directory

$ echo $HOME
/home/user1

Following takes you to user's home directory:
cd $HOME
cd ~
cd

This is an **absolute pathname.** It shows a file's location with reference to the top i.e. root.

pwd: checking your current directory

pwd displays the absolute pathname. It shows the current working directory.

cd: changing the current directory

If prgm1 is present in current directory
$ pwd
/home/user1
$ cd prgm1
$ pwd
/home/user1/prgm1

We can also mention the absolute pathname.
$ pwd
/home/user1/prgm1
$ cd /bin
$ pwd
/bin

We can also use cd without arguments. When we use cd without arguments, it reverts to the home directory.
$ pwd
/home/user1/prgm1
$ cd

$ pwd
/home/user1

mkdir : making directories

We can create directories using mkdir command. The command is followed by names of the directories.

$ mkdir dir1
This will create dir1 directory in the current working directory.

$ mkdir dir11 dir22 dir33
This will create dir11, dir22 and cir33 in the current working directory.

$ mkdir d1 d1/data d1/prgm
This creates three subdirectories- d1 and two subdirectories under d1.

rmdir : removing directories
This command removes the directory. But for this the directory must be empty. And it should be at lower level than current working directory in the hierarchy.
$ rmdir dir1

We can also use this command to delete multiple directories and also hierarchy of directories.

$ rmdir d1/data d1/prgm d1
This must be in reverse order of that used in creating the directories.

Absolute Pathname
If the first character of a pathname is /, the file's location must be determined with respect to root (the first /). Such a pathname is called an **absolute pathname**. No two files in UNIX system can have identical absolute pathnames.

If directory containing the command is specified in the PATH variable then no need to use absolute pathname. If you execute programs residing in some directory that isn't in PATH, the absolute pathname is to be specified. If you are frequently accessing programs in a certain directory, it is better to include the directory in the PATH variable.
Relative Pathnames
Here the pathname does not begin with a /.

Creating a soft link:

ln -s <path-to-other-location> <softlinknamehere>

This is useful whenever we want to refer to some long path location multiple times. Also, whenever we wish to deploy any local folder on the web then we may create softlink to this folder in htdocs.

soft link shows like below:

ll controllers models views

lrwxrwxrwx 1 user1 user1 53 May 29 10:43 controllers ->
/home/user1/web/yiidemo/protected/controllers/

lrwxrwxrwx 1 user1 user1 48 May 29 10:44 models
->/home/user1/web/yiidemo/protected/models/

lrwxrwxrwx 1 user1 user1 47 May 29 10:44 views
->home/user1/web/yiidemo/protected/views/

Contributors (CE Department, DDU):

- Prof. Niyati J. Buch
- Prof. Jigar M. Pandya