Lecture – 13

SECTION -C

Getting Started with UNIX

Absolute & Relative Pathnames:

- Suppose your current working directory is /home/rama
- If there is sub-dir called scripts here then the absolute path for the sub-dir is /home/rama/scripts

and

- relative path is scripts or ./scripts (relative to /home/rama)
- **Absolute path is from home directory to user directory..**

But

- **Relative path is from present working directory to user directory..**

Absolute & Relative Pathnames:

- **Many UNIX commands use file & directory names as arguments, which are presumed to exist in the current directory. For instance the command:**

$ cat login.sql

will work only if the file login.sql exists in your current directory. However, if you are placed in /usr and want to access login.sql in /home/kumar, you can obviously use the command, but rather the pathname of the file :

$ cat /home/kumar/login.sql

- **If the first character of a pathname is /, the file's location must be determined with respect to root (the first /). Such a path name, as above will called an absolute pathname.**
- **When you've more than one / in a pathname, for each such /, you've to descend one level in the file system. Thus kumar is one level below home, and two levels below root.**

- **No two files in a UNIX system can have identical absolute pathnames.**
- **You can have two files with the same name, but in different directories; their pathnames will also be different. Thus the file /home/ kumar/progs/c2f.pl can coexist with the file /home/kumar/safe/c2f.pl**
- **If you execute programs residing in some other directory that isn't in PATH the absolute path needs to be specified. For eg. to execute the program less residing in /usr/local/bin. You need to enter the absolute pathname:**

$ /usr/local/bin/less

If you are frequently accessing programs in a certain directory. Its better to include the directory itself in

Briefs about The PATH:

The sequence of directories that the shell searches to look for a command is specified in its own PATH variable, Use echo to evaluate this variable & you'll see a directory list separated by colons:

```
$ echo $ PATH
/bin: /usr/bin: /usr/local/bin: /usr/ccs/bin:
  /usr/local/java/bin:
```

Relative Pathnames:

- **In previous topic, we didn't use an absolute pathname to move to the directory progs. Nor did we use one as an argument to cat**

$ cd progs

$ cat login.sql

Here, both progs and login.sql are presumed to exist in the current directory. Now, if progs also contain a directory scripts under it, you still won't need an absolute pathname to change to that directory :

$ cd progs/scripts ***progs is in current directory.***

Here we have a pathname that has a /, but it is not an absolute pathname because it doesn't begin with a /.

Using . And .. In relative pathnames:

- You change your directory from /home/kumar/pis/progs to its parent directory (/home/kumar/pis) by using cd with absolute pathname:

$ cd /home/kumar/pis

UNIX offers a shortcut – the relative pathname – that uses either the current or parent directory as reference, and specifies the path relative to it. A relative pathname uses one of these cryptic symbols:

- . (a single dot) – This represents the current directory.
- . . (two dots) – This represents the parent directory.

We'll now use the **. .** to frame relative pathnames. Assuming that you are placed in /home/kumar/progs/data/text, you can use **..** as an argument to cd to move the parent directory,/home/kumar/progs/data

$ pwd

/home/kumar/progs/data/text

$ cd .. *Moves one level up*

$pwd

/home/kumar/progs/data

- This method is compact and more useful when ascending the hierarchy. The command cd .. translates to this : “ **Change your directory to the parent of the current directory.”** You can combine any number of such sets of **. .** separated by **/**s. However, when a / is used with .. it acquires a different meaning; instead of moving down a level, it moves one level up. For instance, to move to /home, you can always use cd /home. Alternatively, you can also use a relative pathname :

$ pwd

/home/kumar/pis

$ cd../.. *Move two levels up*

$ pwd

/home

- **Now lets turn to the single dot that refers to the current directory.**
- **Any command which uses the current directory as argument can also work with a single dot.**
- **This dot is also implicitly included whenever we use a filename as argument, rather than a pathname. For instance,**

$ cd progs **is same as** **$ cd ./progs**

Applications & Research

MS-DOS/Microsoft Windows style

- Contrary to popular belief, the Windows system API accepts slash, and thus all the above Unix examples should work. But many applications on Windows interpret a slash for other purposes or treat it as an invalid character, and thus require you to enter backslash — notably the cmd.exe shell (often called the "terminal" as it typically runs in a terminal window). Note that many other shells available for Windows, such as tcsh and Windows PowerShell, allow the slash.
- In addition "\" does not indicate a single root, but instead the root of the "current disk". Indicating a file on a disk other than the current one requires prefixing a drive letter and colon. No ambiguity ensues, because colon is not a valid character in an MS-DOS filename, and thus one cannot have a file called "A:" in the current directory.
- UNC names (any path starting with \\?\) do not support slashes.[6]
- The following examples show MS-DOS/Windows-style paths, with backslashes used to match the most common syntax:

    A:\Temp\File.txt