Linux Notes (Deepak)

## 1. What is Unix ?

The Unix operating system is a set of programs that act as a link between the computer and the user.

The computer programs that allocate the system resources and coordinate all the details of the computer's internals is called the **operating system** or the **kernel**.

Users communicate with the kernel through a program known as the **shell**. The shell is a command line interpreter; it translates commands entered by the user and converts them into a language that is understood by the kernel.

### System Shutdown

The most consistent way to shut down a Unix system properly via the command line is to use one of the following commands −

|  |  |
| --- | --- |
| Sr.No. | Command & Description |
| 1 | **halt**  Brings the system down immediately |
| 2 | **init 0**  Powers off the system using predefined scripts to synchronize and clean up the system prior to shutting down |
| 3 | **init 6**  Reboots the system by shutting it down completely and then restarting it |
| 4 | **poweroff**  Shuts down the system by powering off |
| 5 | **reboot**  Reboots the system |
| 6 | **shutdown**  Shuts down the system |

## 2. FileSystem ?

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.

## Metacharacters

Metacharacters have a special meaning in Unix. For example, **\*** and **?** are metacharacters. We use **\*** to match 0 or more characters, a question mark (**?**) matches with a single character.

For Example −

$ls ch\*.doc

Displays all the files, the names of which start with **ch** and end with **.doc** −

ch01-1.doc ch010.doc ch02.doc ch03-2.doc

ch04-1.doc ch040.doc ch05.doc ch06-2.doc

ch01-2.doc ch02-1.doc c

Here, **\*** works as meta character which matches with any character. If you want to display all the files ending with just **.doc**, then you can use the following command −

$ls \*.doc

## 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.

## 3. File Permission ?

* **Owner permissions** − The owner's permissions determine what actions the owner of the file can perform on the file.
* **Group permissions** − The group's permissions determine what actions a user, who is a member of the group that a file belongs to, can perform on the file.
* **Other (world) permissions** − The permissions for others indicate what action all other users can perform on the file.

## 4. Pipes & Filters ?

You can connect two commands together so that the output from one program becomes the input of the next program. Two or more commands connected in this way form a pipe.

To make a pipe, put a vertical bar (**|**) on the command line between two commands.

When a program takes its input from another program, it performs some operation on that input, and writes the result to the standard output. It is referred to as a ***filter***.

## The grep Command

The grep command searches a file or files for lines that have a certain pattern. The syntax is −

$grep pattern file(s)

The name **"grep"** comes from the ed (a Unix line editor) command **g/re/p** which means “globally search for a regular expression and print all lines containing it”.

$ls -l | grep "Aug"

-rw-rw-rw- 1 john doc 11008 Aug 6 14:10 ch02

-rw-rw-rw- 1 john doc 8515 Aug 6 15:30 ch07

-rw-rw-r-- 1 john doc 2488 Aug 15 10:51 intro

-rw-rw-r-- 1 carol doc 1605 Aug 23 07:35 macros

$

There are various options which you can use along with the **grep** command −

|  |  |
| --- | --- |
| Sr.No. | Option & Description |
| 1 | **-v**  Prints all lines that do not match pattern. |
| 2 | **-n**  Prints the matched line and its line number. |
| 3 | **-l**  Prints only the names of files with matching lines (letter "l") |
| 4 | **-c**  Prints only the count of matching lines. |
| 5 | **-i**  Matches either upper or lowercase. |

The sort Command

The **sort** command arranges lines of text alphabetically or numerically. The following example sorts the lines in the food file −

$sort food

Afghani Cuisine

Bangkok Wok

Big Apple Deli

Isle of Java

Mandalay

Sushi and Sashimi

Sweet Tooth

Tio Pepe's Peppers

$

The **sort** command arranges lines of text alphabetically by default. There are many options that control the sorting −

|  |  |
| --- | --- |
| Sr.No. | Description |
| 1 | **-n**  Sorts numerically (example: 10 will sort after 2), ignores blanks and tabs. |
| 2 | **-r**  Reverses the order of sort. |
| 3 | **-f**  Sorts upper and lowercase together. |
| 4 | **+x**  Ignores first **x** fields when sorting. |

More than two commands may be linked up into a pipe. Taking a previous pipe example using **grep**, we can further sort the files modified in August by the order of size.

The following pipe consists of the commands **ls**, **grep**, and **sort** −

$ls -l | grep "Aug" | sort +4n

-rw-rw-r-- 1 carol doc 1605 Aug 23 07:35 macros

-rw-rw-r-- 1 john doc 2488 Aug 15 10:51 intro

-rw-rw-rw- 1 john doc 8515 Aug 6 15:30 ch07

-rw-rw-rw- 1 john doc 11008 Aug 6 14:10 ch02

$

This pipe sorts all files in your directory modified in August by the order of size, and prints them on the terminal screen. The sort option +4n skips four fields (fields are separated by blanks) then sorts the lines in numeric order.

## The pg and more Commands

A long output can normally be zipped by you on the screen, but if you run text through more or use the **pg** command as a filter; the display stops once the screen is full of text.

Let's assume that you have a long directory listing. To make it easier to read the sorted listing, pipe the output through **more** as follows −

$ls -l | grep "Aug" | sort +4n | more

-rw-rw-r-- 1 carol doc 1605 Aug 23 07:35 macros

-rw-rw-r-- 1 john doc 2488 Aug 15 10:51 intro

-rw-rw-rw- 1 john doc 8515 Aug 6 15:30 ch07

-rw-rw-r-- 1 john doc 14827 Aug 9 12:40 ch03

.

.

.

-rw-rw-rw- 1 john doc 16867 Aug 6 15:56 ch05

--More--(74%)

The screen will fill up once the screen is full of text consisting of lines sorted by the order of the file size. At the bottom of the screen is the **more** prompt, where you can type a command to move through the sorted text.

Once you're done with this screen, you can use any of the commands listed in the discussion of the more program.

## 5. Regular Expression ?

A regular expression is a string that can be used to describe several sequences of characters. Regular expressions are used by several different Unix commands, including **ed**, **sed**, **awk**, **grep**, and to a more limited extent, **vi**.

Here **SED** stands for **s**tream **ed**itor. This stream-oriented editor was created exclusively for executing scripts. Thus, all the input you feed into it passes through and goes to STDOUT and it does not change the input file.

* SED is a powerful text stream editor. Can do insertion, deletion, search and replace(substitution).
* SED command in unix supports regular expression which allows it perform complex pattern matching.

**Syntax ::**

sed OPTIONS... [SCRIPT] [INPUTFILE...]

1. **Replacing or substituting string :** Sed command is mostly used to replace the text in a file. The below simple sed command replaces the word “unix” with “linux” in the file.

sed 's/unix/linux/' geekfile.txt

2. **Replacing the nth occurrence of a pattern in a line :** Use the /1, /2 etc flags to replace the first, second occurrence of a pattern in a line. The below command replaces the second occurrence of the word “unix” with “linux” in a line.

sed 's/unix/linux/2' geekfile.txt

3. **Replacing all the occurrence of the pattern in a line :** The substitute flag /g (global replacement) specifies the sed command to replace all the occurrences of the string in the line.

sed 's/unix/linux/g' geekfile.txt

4. **Replacing from nth occurrence to all occurrences in a line :** Use the combination of /1, /2 etc and /g to replace all the patterns from the nth occurrence of a pattern in a line. The following sed command replaces the third, fourth, fifth… “unix” word with “linux” word in a line.

$sed 's/unix/linux/3g' geekfile.txt

5. **Parenthesize first character of each word :** This sed example prints the first character of every word in parenthesis.

echo "Welcome To The Geek Stuff" | sed 's/\(\b[A-Z]\)/\(\1\)/g'

**Output:**

(W)elcome (T)o (T)he (G)eek (S)tuff

## 6. User Administration ?

## Managing Users and Groups

There are four main user administration files −

* **/etc/passwd** − Keeps the user account and password information. This file holds the majority of information about accounts on the Unix system.
* **/etc/shadow** − Holds the encrypted password of the corresponding account. Not all the systems support this file.
* **/etc/group** − This file contains the group information for each account.
* **/etc/gshadow** − This file contains secure group account information.