Lab 1: Frequent Commands and Tools in Linux

1.Objective

- Study Linux commands which are frequently used.

2.Syllabus

- Frequent Linux commands
- Linux tools

3.Prerequisite

- Preview Linux introduction
- Computer which run Linux system (e.g. Ubuntu)

4.Contents

4.1 Overview of Shell

(1) ***What Is "The Shell"?***

Simply put, the shell is a program that takes commands from the keyboard and gives them to the operating system to perform. In the old days, it was the only user interface available on a Unix-like system such as Linux. Nowadays, we have graphical user interfaces (GUIs) in addition to command line interfaces (CLIs) such as the shell.

On most Linux systems a program called bash (which stands for Bourne Again SHell, an enhanced version of the original Unix shell program, sh, written by Steve Bourne) acts as the shell program. Besides bash, there are other shell programs that can be installed in a Linux system. These include: ksh, tcsh and zsh.

(2) ***What's A "Terminal?"***

It's a program called a terminal emulator. This is a program that opens a window and lets you interact with the shell. There are a bunch of different terminal emulators you can use. Most Linux distributions supply several, such as: gnome-terminal, konsole, xterm, and eterm.

(3) ***Starting A Terminal***

Your window manager probably has a way to launch a terminal from the menu. Look through the list of programs to see if anything looks like a terminal emulator. If you are a KDE user, the terminal program is called "konsole," in Gnome it's called "gnome-terminal." You can start up as many of these as you want and play with them. While there are a number of different terminal emulators, they all do the same thing. They give you access to a shell session. You will probably develop a preference for one, based on the different bells and whistles each one provides.

4.2 Frequent Linux Commands

There are many build-in commands in Linux system. And some extra commands are also extended. In this experiment, we will learn some basic Linux commands. Note that, if you want to understand more detailed information of commands, you can use the following command (“$” is a shell prompt):

$**man <command>**

Example: **man ls**

Now, let’s learn Linux commands. If you have any question, please don’t hesitate to ask me.

(1) ***pwd***

Think of the file system tree as a maze, and you are standing in it. At any given moment, you are located in a single directory. Inside that directory, you can see its files and the pathway to its parent directory and the pathways to the subdirectories of the directory in which you are standing. The directory you are standing in is called the *working directory*. To find the name of the working directory, use the ***pwd*** command.

Example: $**pwd**

(2) ***ls***

When you first log on to a Linux system, the working directory is set to your home directory. This is where you put your files. On most systems, your home directory will be called ***/home/your_user_name***, but it can be anything according to the whims of the system administrator. To list the files in the working directory, use the ***ls*** command.

Example: $**ls, $ls -a, $ls -al, $ls -acl**

(3) ***cd***

To change your working directory (where you are standing in the maze) you use the ***cd*** command. To do this, type cd followed by the pathname of the desired working directory. A pathname is the route you take along the branches of the tree to get to the directory you want. Pathnames can be specified in one of two different ways: absolute pathnames or relative pathnames.

Example: $**cd /usr/bin, $cd ../, $cd ../<relative directory>, $cd ~**

(4) ***touch***

The **touch** sets the modification and access times of files. If any file does not exist, it is created with default permissions. By default, touch changes both modification and access times. The **-a** and **-m** flags may be used to select the access time or the modification time individually. Selecting both is equivalent to the default. By default, the timestamps are set to the current time. The **-t** flag explicitly specifies a different time, and the **-r** flag specifies to set the times those of the specified file. The **-A** flag adjusts the values by a specified amount.

Example: $**touch test.txt**

(5) ***cat***

The **cat** concatenates files, or the standard input, to the standard output. It lists contents of files or the standard input.

Example: $**cat test.txt**

(6) ***find***

The **find** searches the directory tree rooted at each given file name by evaluating the given expression from left to right, according to the rules of precedence, until the outcome is known, at which point find moves on to the next file name.

Example: $**find < pathname> -name "*.log"**

(7) ***head***

The **head** prints the first 10 lines (or indicate by **-n**) of each file to standard output. With more than one file, precede each with a header giving the file name. With no file, or when the file is **-**, read standard input.

Example: $**head test.txt -n 100**

(8) ***tail***

The **tail** prints the last 10 lines (or indicate by **-n**) of a file to standard output. With more than one file, precede each with a header giving the file name. With no file, or when file is **-**, read standard input.

Example: $**tail test.txt -n 100**

(9) ***cp***

The **cp** command is used to make copies of files and directories. It copies SOURCE to DEST, or multiple SOURCE(s) to DIRECTORY.

Example: $**cp -R /one/two /three/four,** $**cp file1 file2,** $**cp file /one/**

(10) ***mv***

The **mv** command is used to move or rename files. It renames SOURCE to DEST, or move SOURCE(s) to DIRECTORY.

Example: $**mv file1 file2,** $**mv file <path to a directory>**

(11) ***rm***

The **rm** removes each specified file. By default, it does not remove directories. If the **-I** option is given, and there are more than three files or the **-R** is given, then the **rm** prompts the user for whether to proceed with the entire operation.

Example: $**rm file,** $**mv -R <path to a directory>**

(12) ***mkdir***

The **mkdir** creates directories, if they do not already exist.

Example: $**mkdir <directory name>**

(13) ***>, >>, |***

The ">" and ">>" are two redirected methods. Therein, the former rewrite a file and the latter appends information to a file. The "|" means a pipe.

Example: $**cat file1 > file2,** $**cat file1 >> file2,** $**cat file | <other action>**

(14) ***grep***

The **grep** searches the named input files (or standard input) for lines containing a match to the given pattern. By default, the **grep** prints the matching lines.

Example: $**cat file | grep <pattern>**

(15) ***wc***

The **wc** prints newline, word, and byte counts for each file, and a total line if more than one file is specified. With no file, or when file is **-**, read standard input. A word is a non-zero-length sequence of characters delimited by white space.

Example: $**wc -lwc file**

(16) ***tar***

The **tar** stores and extracts files from a compressed package (suffix with "tar.gz, tar") or disk archive.

Example: $**tar -zcvf file.tar.gz <source file or directory>,** $**tar -zxvf file.tar.gz**

(17) ***netstat***

The **netstat** prints information about the Linux networking subsystem. By default, the **netstat** displays a list of open sockets. If you don't specify any address families, then the active sockets of all configured address families will be printed.

Example: $**netstat -s,** $**netstat -i eth0**

(18) ***chmod***

The **chmod** change the mode of a file to a specified mode. The operator **+** causes the selected file mode bits to be added to the existing file mode bits of each file. And the operator **-** causes them to be removed. [r: 4, w: 2, x: 1, rw: 6, rx: 5, wx: 3, rwx: 7]

Example: $**chmod ugo+r file,** $**chmod u+x file**

(19) ***chown***

The **chown** changes the user and/or group ownership of each given file. If only an owner is given, that user is made the owner of each given file, and the file's group is not changed. If the owner is followed by a colon and a group name, with no spaces between them, the group ownership of the file is changed as well. If a colon but no group name follows the user name, that user is made the owner of the file and the group of the file is changed to that user's login group. If the colon and group are given, but the owner is omitted, only the group of the files is changed. If only a colon is given, or if the entire operand is empty, neither the owner nor the group is changed.

Example: $**chown -R <user>:<group> file**

(20) ***which***

The **which** returns the pathnames of the file which would be executed in the current environment. It does this by searching the PATH environment variable for executable files matching the names of the arguments. It does not follow symbolic links.

Example: $**which cat,** $**which wc,** $**which which**

(21) ***ps***

The **ps** displays information about a selection of the active processes. By default, the **ps** selects all processes with the same effective user ID as the current user and associated with the same terminal as the invoker. It displays the process ID, the terminal associated with the process, the cumulated CPU time in hh:mm:ss format, and the executable name. Output is unsorted by default.

Example: $**ps**

(22) ***kill***

The default signal for kill is TERM. Use **-l** or **-L** to list available signals. Particularly useful signals include HUP, INT, KILL, STOP, CONT, and 0. Alternate signals may be specified in three ways: **-9**, -SIGKILL or -KILL. Negative PID values may be used to choose whole process groups; see the PGID column in **ps** command output. A PID of **-1** is special; it indicates all processes except the kill process itself and init.

Example: $**kill -9 -1,** $**kill 123**

4.3 Frequent Linux Tools

(1) ***gdb***

The GNU Debugger, usually called just GDB and named **gdb** as an executable file, is the standard debugger for the GNU operating system.

Example: **$gdb program**

(2) ***ldd***

The **ldd** prints the shared libraries required by each program or shared library specified on the command line.

Example: **$ldd /bin/bash**

(3) ***lsof***

The **lsof** lists open files. Open files in the system include disk files, named pipes, network sockets and devices opened by all processes.

Example: **$lsof**

(4) ***pstack***

The **pstack** prints a stack trace of running processes. The pstack attaches to the active processes named by pids on the command line, and prints out an execution stack trace, including a hint at what the function arguments are.

Example: **$pstack 345**

(5**)** ***strace***

The **strace** traces system calls and signals.

Example: **$strace ls**

(6) ***ipcs***

The **ipcs** provides information on the ipc facilities for which the calling process has read access. The **-i** option allows a specific resource id to be specified. Only information on this id will be printed.

Example: **$ipcs**

(7) ***top***

The **top** program provides a dynamic real-time view of a running system. It can display system summary information as well as a list of processes or threads currently being managed by the Linux kernel.

Example: **$top**

(8) ***free***

The **free** displays the total amount of free and used physical and swap memory in the system, as well as the buffers used by the kernel.

Example: **$free**

(9) ***vmstat***

The **vmstat** reports information about processes, memory, paging, block IO, traps, disks and cpu activity.

Example: **$vmstat**

(10) ***iostat***

The **iostat** command is used for monitoring system input/output device loading by observing the time the devices are active in relation to their average transfer rates. The iostat command generates reports that can be used to change system configuration to better balance the input/output load between physical disks.

Example: **$iostat**

(11) ***wget***

GNU Wget is a free utility for non-interactive download of files from the Web. It supports HTTP, HTTPS, and FTP protocols, as well as retrieval through HTTP proxies.

Example: **$wget http://cn.wordpress.org/wordpress-3.1-zh_CN.zip**

(12) **nmon**

The nmon is a systems administrator, tuner, benchmark tool. It can display the CPU, memory, network, disks (mini graphs or numbers), file systems, NFS, top processes, resources (Linux version & processors) and on Power micro-partition information.

Example: **$nmon**

(13) ***iftop***

The **iftop** listens to network traffic on a named interface, or on the first interface it can find which looks like an external interface if none is specified, and displays a table of current bandwidth usage by pairs of hosts.

Example: **$iftop**

(14) ***htop***

The htop is a free ncurses-based process viewer for Linux. It is similar to top, but allows you to scroll vertically and horizontally, so you can see all the processes running on the system, along with their full command lines. Tasks related to processes (killing, renicing) can be done without entering their PIDs.

Example: **$htop**

5.Conclusion

In the experiment, we must have understood some basic and frequent commands and tools in Linux system.

6.Reference

[1] Learning The Shell. http://linuxcommand.org/lc3_learning_the_shell.php
[2] UNIX Tutorial. http://www.tutorialspoint.com/unix/index.htm