LINUX Shell Programming

Part 1

Presented by:

Marie Luvett S. Interino-Goh, MIT

Topics Outline:

- Quick Introduction to Linux
- Linux Commands
- Getting Started with Shell Programming
- Shell Programming Exercises

Quick Introduction to Linux

Topics:

- What is Linux?
- Who developed the Linux?
- How to get Linux?
- Where can I use Linux?
- Major Components of Linux

Quick Introduction to Linux

- **What is Linux?**
  - Free
  - Unix Like
  - Open Source
  - Network operating system

Quick Introduction to Linux

- **Who developed Linux?**
  - Linus Torvalds
  - University of Helsinki in Finland
  - 1991
  - He used special educational experimental purpose operating system called **Minix** (small version of Unix and used in Academic environment). But due to Minix limitations. Linus felt he could do better than the Minix. So he developed his own version of Minix, which is now know as **Linux.**

Quick Introduction to Linux

- **How to get Linux?**
  - Download over the net
  - Order CD from Linux distributors
  - Linux Distributions are as follows:

redhat

CALDERA systems

SuSE

slackware linux

MandrakeSoft™

debian

Quick Introduction to Linux

- **Where can I use Linux?**
  - Server OS
  - Stand-alone OS

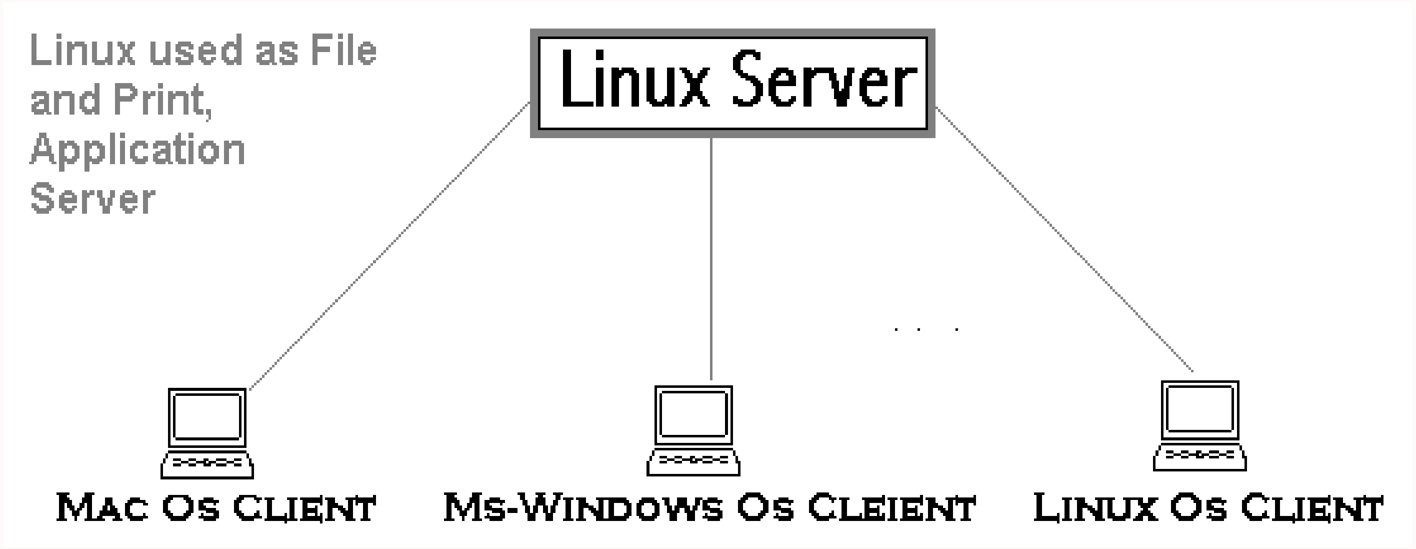
As a server OS, it provides different services/network resources to client. Server OS must be:

- Stable
- Robust
- Secure
- High Performance

Quick Introduction to Linux

- **Linux as Server OS**

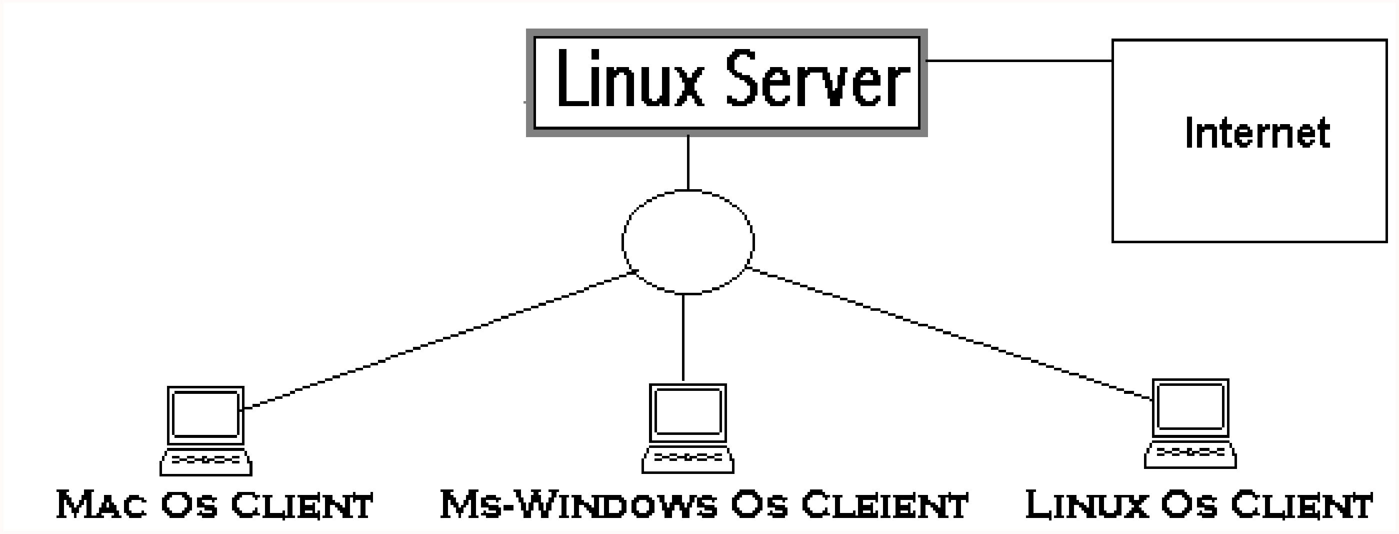
  Linux offers all of the mentioned characteristics plus its **Open Source** and **Free OS**.



(A) Linux Server with different Client OS

Quick Introduction to Linux

- **Linux as Server OS**



(B) Linux Server can act as Proxy/Mail/WWW/Router Server etc.

Quick Introduction to Linux

- **Linux as Stand-alone OS**

  Linux offers **bundled applications** such as follows:

  - Open Office (Writer, Impress, Calc)
  - Graphics/image-editing software (Gimp)
  - Software development (Python)
  - Internet, e-mail, chatting
  - Small personal database management system, etc.

•

Quick Introduction to Linux

- **Major Components of Linux**
  - Kernel
  - Shell
  - File System
  - Communication/Networking
  - Text Processing
  - Programming
  - System Management
  - Online Documentation
  - Graphical Environment

Linux Commands

Topics:

- Directory Commands
- File Readers
- File Operation Commands
- Redirection of Standard Output/Input
- Data Refinement
- File Permission

Linux Commands

- **Directory Commands**
  - To displays the contents of the current working directory

    Syntax:

    ls

Options that can be used with it.

- -a
- -A
- -d
- -l
- -r
- -R

Linux Commands

- **Directory Commands**
  - To change directory

    Syntax:
    - **cd .** means the current directory
    - **cd ..** means parent directory
    - **cd** means will take you to your home directory
    - **cd –** will take you to your previous directory
    - **cd ~** username will take you to the home directory of the user
    - **cd <directory_name>**
    - **cd <directorypath>**

Linux Commands

- **Directory Commands**
  - To make directory/directories

    Syntax:

    mkdir [-option] directory1 directory2 …

    Examples:

    - To create directories dir1, dir2, dir3, on the current directory:

      Syntax: **mkdir dir1 dir2 dir3**

      mkdir my\ folder

    - To create the directory /home/tester/mydir/testdir (mydir is not yet existing):

      Syntax: **mkdir –p /home/tester/mydir/testdir**

Linux Commands

- **Directory Commands**
  - To remove directory

    Syntax:

    rmdir [-option] directoryname

    **This command allows removing EMPTY directories.*

    Examples:

    - To remove the directory dir1

      Syntax: **rmdir dir1**

    - To remove the directory /home/tester/mydir/testdir and its parent directory

      Syntax: **rmdir -p mydir/testdir**

Linux Commands

- <u>**Directory Commands**</u>
  - To remove directory that is not empty

    Syntax:

    rm -r directoryname

    rm -rf directoryname

    Examples:

    - To remove the directory and subdirectories of dir1

      Syntax: **rm –r dir1**

Linux Commands

- **Directory Commands**
  - To print or display the current working directory.

    Syntax: **pwd**
  - To clear the screen.

    Syntax: **clear**
  - To display previously entered commands. This information is stored the ~/.bash_history file located at the home directory of each user.

    Syntax: **history**

Linux Commands

- **File Readers** allow to view the contents of a file.
  - To concatenates a file /files and displays the output on the screen.

    Syntax: **cat [filename1] [filename2]**

    Examples:
  - To view the contents of a file phone1:

    Syntax: **cat phone1**
  - To view the contents of both phone1 and phone 2:

    Syntax: **cat phone1 phone2**

Linux Commands

- **File Readers**
  - The **more** command pages through the text of a file “one screen at a time”.

    Press **<Spacebar>** to view the next page

    Syntax: **more [filename1]**
  - The **less** command navigates through the file. Can go down or go back to the previous page.

    Syntax: **less [filename1]**

Note:

After viewing the file, press **q** to quit the *lesser* environment

Linux Commands

- **File Readers**
  - The **head** command displays the first lines of a file

    Syntax: **head [-count] filename**

    Examples:

    - To view the first 10 lines:

      Syntax: **head song.txt**
    - To view the first 5 lines:

      Syntax: **head-5 song.txt**

Linux Commands

- **File Readers**
  - The **tail** command displays the last lines of a file.

    Syntax: **tail [-count] filename**

    Examples:
    - To view the last 10 lines:

      Syntax: **tail song.txt**
    - To view the last 5 lines:

      Syntax: **tail -5 song.txt**

- **File Readers**
  - The **wc** command allows you to count the number of lines, words, and characters in a file.

    Syntax: **wc [-option] filename**

    Examples:
    - To count the number of lines, words, and characters in the file *list*:

      Syntax: **wc list**
    - To count the number of lines in the file *list***:**

      Syntax: **wc –l list**
    - To count the number of words in the file *list*:

      Syntax: **wc –w list**
    - To count the number of characters in the file *list*:

      Syntax: **wc –c list**

Linux Commands

- **File Readers**
  - The **man** command means manual. It displays description of a certain command

    Syntax: **man command**

    Examples:

    - To view the manual for **ls** command

      Syntax: **man ls**

Note: *To exit in the* **man** *page, press* **q**.

Linux Commands

- **File Operation Commands**
  - The **cp** command allows copying file/s and directories from one location to another.

    Syntax: **cp source destination**

    Examples:
    - To copy phone1 to directory dir1:

      Syntax: **cp phone1 dir1**
    - To copy dir1 and its contents to directory dir2:

      Syntax: **cp –R dir1 dir2**

Linux Commands

- **File Operation Commands**
  - The **mv** command allows to move and rename files

    Syntax: **mv oldname newname**

    mv source destination

    Examples:
    - To rename the file phone1 to phone3:

      Syntax: **mv phone1 phone3**
    - To move the file phone2 to dir 2:

      Syntax: **mv phone2 dir2**
    - To move the directory dir2 to the directory dir3:

      Syntax: **mv dir2 dir3**

Linux Commands

- **File Operation Commands**
  - The **touch** command allows creating an empty file. It also allows updating the time stamp on existing file.

  Syntax: **touch filename1 filename2**

  Examples:
  - To create an empty files aa, bb and cc:

    Syntax: **touch aa bb cc**
  - To update the time stamp of the file song.txt:

    Syntax: **touch song.txt**

Linux Commands

- **File Operation Commands**
  - The **touch** command can also create hidden files

  Syntax: **touch .filename1 .filename2**

  Examples:

  - To create an empty hidden file ***sample***:

    Syntax: **touch .sample**

Linux Commands

- **File Operation Commands**
  - The **rm** command will delete a file forever.

    Syntax: **rm [-option] filename1 filename2 …**

    Examples:

    - To delete the file aa:

      Syntax: **rm aa**
    - To prompt first before removing the file bb:

      Syntax: **rm –I bb**
    - To remove all files and all sub-directories and their contents:

      Syntax: **rm -r ***
    - To remove forcefully all files and all sub-directories and their contents:

      Syntax: **rm -rf ***

Caution: Exercise caution when executing the last two commands. Should you exercise either of them from the root directory (/), your system will definitely crash.

Linux Commands

- **File Operation Commands**
  - The **echo** command displays the string or text specified after it. It also used to reference and display the values of variables. It is commonly used in programs, or shell scripts, were user input is needed.

    Syntax: **echo [string]**

    echo $variablename

    Examples:

    - To re-echo the word "hello" on the command line:

      Syntax: **echo hello**

    - To display the value of the variable "x"

      Syntax: **x=hello**

      Syntax: **echo $x**

Linux Commands

- **File Operation Commands**
  - The **cmp** command checks two files to see if they differ. It does a byte-by-byte comparison of file1 and file2. If the files differ, cmp outputs the location at which the first difference occurs.

    Syntax: **cmp [options] file1 [file2]**

    Examples:

    cat a: The quick brown fox jumped over the lazy dog's back.
    cat b: The quick brown fox jumped over the lasy dog's back.

    - To check whether file a and b differ to each other:

      Syntax: **cmp a b**

      a b differ: char 39, line 1

Linux Commands

- **File Operation Commands**
  - The **file** command determines the file type of a given file. It reads the first few bytes of a file to determine the file type.

    Syntax: **file [filename]**

    Examples:
    - To display the file type of the *list* file:

      Syntax: **file list**

Linux Commands

- **Redirection Standard Input/Output**

  There are three main redirection symbols **>, >>, <**

  - Redirector Symbol **>**

    Syntax: **Linux-command > filename**

  To output Linux-commands result (output of command or shell script) to file. Note that if file already exist, it will be overwritten else new file is created.

  Example:

  - To send output of ls command give:

    Syntax: **ls > myfiles**

  Now if '**myfiles**' file exist in your current directory it will be overwritten without any type of warning.

Linux Commands

- **Redirection Standard Input/Output**
  - Redirector Symbol **>>**

    Syntax: **Linux-command >> filename**

To output Linux-commands result (output of command or shell script) to END of file. Note that if file exist , it will be opened and new information/data will be written to END of file, without losing previous information/data, And if file is not exist, then new file is created.

Example:

- To send output of date command to already exist file give command

  Syntax: **date >> myfiles**

Now if '**myfiles**' file exist in your current directory it will be overwritten without any type of warning.

Linux Commands

- **Redirection Standard Input/Output**
  - Redirector Symbol **<**

    Syntax: **Linux-command < filename**

To take input to Linux-command from file instead of keyboard.

Example:

- To take input for cat command give

  Syntax: **cat < myfiles**

Now if '**myfiles**' file exist in your current directory it will be overwritten without any type of warning.

Linux Commands

- **Redirection Standard Input/Output**

  Examples:

  - You can also use above redirectors simultaneously as follows:

  Create text file sname as follows:

  Syntax: **cat > sname**

  ```
  vivek
  ashish
  zebra
  babu
  ```

  Press CTRL + D to save.

  Now issue following commands:

  Syntax: **sort < sname > sorted_names**

  cat sorted_names

  ```
  ashish
  babu
  vivek
  zebra
  ```

Linux Commands

- **Redirection Standard Input/Output**
  - Pipe Symbol **|**

    A **pipe** has the same as redirecting standard output. It is nothing but a temporary storage place where the output of one command is stored and then passed as the input for second command. Pipes are used to run more than two commands (multiple commands) from same command line.

Syntax: **command1 | command2**

Example:

This command line uses a pipe to generate the same result as the following group of command lines:

Syntax:

```
sort  sname | cat > sorted_names
cat sorted_names
ashish
babu
vivek
zebra
```

Linux Commands

- **Redirection Standard Input/Output**

Command using Pipes	Meaning or Use of Pipes
$ ls | more	Output of ls command is given as input to more command So that output is printed one screen full page at a time.
$ who | sort	Output of who command is given as input to sort command So that it will print sorted list of users
$ who | sort > user_list	Same as above except output of sort is send to (redirected) user_list file
$ who | wc -l	Output of who command is given as input to wc command So that it will number of user who logon to system
$ ls -l | wc -l	Output of ls command is given as input to wc command So that it will print number of files in current directory.
$ who | grep raju	Output of who command is given as input to grep command So that it will print if particular user name if he is logon or nothing is printed

Linux Commands

- **Data Refinement Commands**
  - The **sort** command sorts and/or merges one or more text files in sequence.

Syntax: **sort filename**

Example:

Syntax: **cat days**

```
Monday
Tuesday
Wednesday
Thursday
Friday
```

sort days

```
Friday
Monday
Thursday
Tuesday
Wednesday
```

Linux Commands

- **Data Refinement Commands**
  - The **uniq** command displays a file, removing all but one copy of successive repeated lines. If the file has been sorted, **uniq** ensures that no two lines that it displays are the same.

Syntax: **uniq filename**

Example:

Syntax:

```
cat f1
apple
apple
banana
banana

uniq f1
apple
banana
```

Linux Commands

- **Data Refinement Commands**
  - The **grep** command is primarily for pattern searching.

Users can use this command to search a set of files for one or more phrases or patterns. If the pattern exists, then grep will print all the lines that contain the said pattern.

Syntax: **grep pattern filename**

where:

- **pattern** is the phrase or pattern the user wants to find.
- **filename** is the name of the target file.

Linux Commands

- **Data Refinement Commands**
  - The **grep** command

Assume that the file ***horror.story*** contains:

And he slowly entered the cell and went to the sink.

He reached out trembling and touched the lather on a brush.

It was real. It felt warm. It smelled of soap.

Example:

```
grep the horror.story
And he slowly entered the cell and went to the sink.
He reached out trembling and touched the lather on a brush.
$ _
```

Linux Commands

- **Data Refinement Commands**
  - The **grep** command
    - ✓ If the pattern does not exist, then UNIX will simply display the $ prompt again.
    - ✓ Notice that the third line matched since the is part of lather.
    - ✓ If the pattern consists of more than one word, then the user must enclose the pattern in double quotes.

Example:

```
grep “of soap” horror.story
It was real. It felt warm. It smelled of soap.
```

Linux Commands

- **Data Refinement Commands**
  - The **grep** command

    Some of the options available for the grep command are:
    - The **-v** Option

    The -v option will display all lines except those containing the pattern

    Example:

    ```
    grep -v "of soap" horror.story
    ```

    ```
    And he slowly entered the cell and went to the sink.
    He reached out trembling and touched the lather on a brush.
    It was real. It felt warm.
    ```

Linux Commands

- **Data Refinement Commands**
  - The **grep** command

    Some of the options available for the grep command are:
    - The **-n** option will display the lines together with their line number.

Example:

```
grep -n "of soap" horror.story
4: It smelled of soap.
```

Linux Commands

- **File Permission Commands**

The **chmod** command allows changing the file access permission of a file.

Syntax: **chmod mode filename**

Techniques:

- **Symbolic Mode**
- **Absolute Mode**

Linux Commands

- **File Permission Commands**
  - **Symbolic Mode**

    The FIRST set determines who is granted or denied a specific set of permissions. The first 3 sets of flags are as follows:

    - u= user/owner of the file
    - g=users belonging to the same group as the file's group set
    - o= other users
    - a= all (owner, group and others)

    The SECOND set of flags determines whether permissions will be added, removed, or set:

    - + ------ add permission
    - - ------ remove permission
    - = ------ set permission

Linux Commands

- **File Permission Commands**
  - **Symbolic Mode**

The THIRD set determines what permissions will be given

- r = read
- w = write
- x = execute

Examples:

- To change the permission of *file1* from –rw-r- -r- - to –rw-rw-rw- :

  Syntax: **chmod go+w file1**

- To change the permission of *file1* from –rw-rw-rw- to –rw-rw-r-- :

  Syntax: **chmod u-w file1**

- To change the permission of *file1* from –rw-rw-r to –rwxrw-rw- :

  Syntax: **chmod u+x file1 | chmod o+w file1**

  chmod u=rwx | chmod go=rw file1

Linux Commands

- **File Permission Commands**
  - **Aboslute Mode** / **Octal Mode** changes a file's permission by using numbers/octal notation. The numeric mode is the sum of one or more of the following values:
    - **r =4**
    - **w =2**
    - **x =1**

  Examples:

  - To change the permission of *file1* from –rw-r- -r- - to –rw-rw-rw-:

    Syntax: **chmod 666 file1**
  - To change the permission of *file1* from –rw-rw-rw- to –rw-rw-r-- :

    Syntax: **chmod 664 file1**
  - To change the permission of all the files in the directory dir1:

    Syntax: **chmod –R 755 dir1**

End of Part 1

Thank you.