Linux Foundation

Basic Commands and Administrative Commands

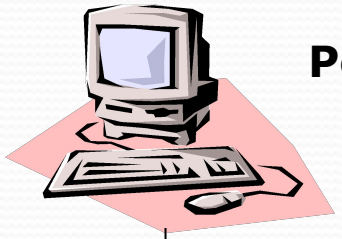
EduRamp ELs
The B2B Learning Specialist

Objectives

At the end of this module you will learn about:

- Introduction to bash shell
- Getting Started - Shell prompts, Linux Command - arguments & options
- Printing the working directory - pwd
- Getting help on commands
- Managing files & directories
- Hard link & Soft Link
- find command - to search for files and directories
- Filters - grep
- File system commands – df & du
- The awk Programming Language
- Working with run levels

System boot up



Power on

Kernel get loaded

File system get mounted

init Process

/etc/inittab

login

Virtual Console

- Though you may not have more than one physical console (a monitor plus a keyboard) connected to your PC, you can use virtual consoles to log in simultaneously to more than one account on your system.

- When any Linux system first boots up, you get a normal Linux login: prompt, so you can log in, and start X11 or do whatever else you would do with a single Linux shell.

- Linux is pretty much the same, except that instead of just one such login:, you get several. These are accessed using the key combinations CTrl +<ALT><F1>, <ALT><F2>, <ALT><F3> etc.

- Each of the "screens" with you see is known as a "Virtual Console".

System Directories

- **Find below few important system directories**

/ : Root directory.

/bin : Command-line executable directory.

/dev : Device directory

/etc : System configuration files and executable directory.

/lib : The library directory.

/home : Consists of the users' home directories

/usr : Linux System Resources directory

Introduction to bash shell

- bash is a Linux shell written by Brian Fox for the GNU project as a free software replacement for the Bourne shell.
- Released in 1989, it has been distributed widely as the shell for the GNU operating system and as the default shell on Linux, Mac OS X and Darwin.
- As an acronym, it stands for Bourne-again shell, referring to its objective as a free replacement for the Bourne shell.

Getting Started

- Once the system is connected to Linux Server, a user is prompted for a login user id and password.
- The login userid is a unique id on the system. The password is changeable code known only to the user.
- At the login prompt, the user should enter the user id; at the password prompt, the current password should be typed.

- **Note:** Linux is case sensitive. Therefore the login and password should be typed exactly as issued; the login, at least, will normally be in lower case.

The shell prompts

- Once you login successfully, the shell prompt appears at the left side of your screen

- This prompt means the system is waiting for you to type in some Linux command.

- **Note:** Generally the shell prompt of $ indicate a regular user login and the prompt of # indicates a root user login.

Linux Command – Arguments & Options

- **Syntax of a Linux Command:**

  command [options] [arguments]

- **See some examples:**

```
ls  -l              # command with one option
ls -l -a 'or' ls -la    # Command with multiple options

ls  /etc            # command with one argument
ls  /etc  /dev          # command with multiple arguments
```

Print the working directory - pwd

- **pwd**
  - Displays the current working directory.

Getting Help on Commands

- The Linux manual, usually called man pages, is available to explain the usage of the Linux system and commands.

- **Syntax:**
  man [options] command_name
- **Common Options**
  - -k keyword list command synopsis line for all keyword matches
  - -M path path to man pages
  - -a show all matching man pages (SVR4)
- info command_name => gives information about commands
- command_name --help =>gives command syntax

touch

- The command **touch** is used to change the time stamp of the file

- **Syntax:**
  touch [options] file
- **Options:**
  -a to change the access time
  -m to change the modification time
  -c no create if the file do not exists
- touch <file> will change the timestamps of the file if the file exists
- If the file does not exist, it will create a file of zero byte size.

Listing Files

- The command **ls** is used to list the names of files and directories.

- **Syntax:**
  ls [options] [file….]

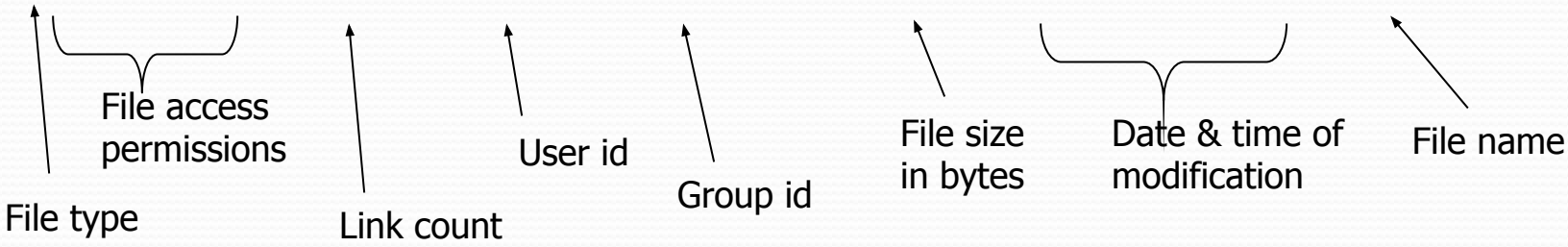
- **options:**

-l	list in long format
-a	list all files including those beginning with a dot
-i	list inode no of file in first column
-s	reports disk blocks occupied by file
-R	recursively list all sub directories
-F	mark type of each file
-C	display files in columns

Listing the Directory Contents

$ **ls –l**

```
lrwxr-xr-x    1    user1    projA    12373    Dec  15  14:45  a.out
drwxr-xr-x    2    user2    projD     4096    Dec  22  14:00  awkpro
brw-r--r--    1    user1    projA    12831    Dec  12  13:59  c
crw-------    1    user1    projA    61440    Dec  15  11:16  core
-rw-r--r--    1    user3    projC      255    Dec  20  14:29  cs
```



Meta Characters

Char	Meaning	Example	Possible Output
*	Match with zero or multiple number of any character.	$ ls –l *.c file*	prog1.c, prog2.c, file1 , file2, filebc
?	Match any Single character	$ ls –l file?	filea , fileb, file1
[..]	Match with any single character with in the bracket	$ ls –l file[abc]	filea, fileb,filec
;	Command separator	$cat filea; date	displays the content of filea and displays the current date and time
|	Pipe two commands	$ cat filea | wc -l	Prints the number of lines of filea
()	Group commands, used when the output of the command group has to be redirected	$ (echo “***x.c***”;cat x.c)>out	Redirects the content of x.c with a heading ***x.c*** to the file out

Path names – Absolute & Relative

- The **path** of a file can be represented by either absolute path or relative path.

- **Absolute path:**

  Always begin with the root directory (/). It's a complete roadmap to the file location.

  eg:

  ls –l /home/user1/file1

- **Relative Path:**

  Describes the location of a file/directory with reference to the current directory.

  eg:

  ls –l user1/file1 => assuming that your present working directory is /home, this represents the file /home/user1/file1

Directory Creation

- **The command mkdir create directory**

- **Syntax:**

  mkdir [option] <directory name>

  $ mkdir <path>/<directory>

  $mkdir –p <directory1>/<directory2>/<directory3>

- **Example:**

  $ mkdir project1 =>This creates a directory project1 under current directory

- **Note:** Write and execute permissions are needed for the directory in which user wants to create a directory

Directory Removal

- The command **rmdir** removes directory

- **Syntax**
  rmdir <directory name>

- **Example**
  rmdir project1 => Removes project1 directory in the current directory
  rmdir dir1 dir2 => Remove multiple directories
  rmdir –p dir1/dir2/dir3 => Remove the directory hierarchy

- **Note:** rmdir removes a directory if it is empty and is not the current directory

Changing Directories

- The **cd** command is used to change the directory

cd => take to the home directory

cd .. => takes to the parent directory

cd / => takes to the root directory

File-Related Commands

File Operation	Command
Copying a file	cp
Moving a file	mv
Renaming a file	mv
Removing a file	rm
Displaying a file and concatenating files	cat

Command for copying - cp

- The command **cp** is used to copy files across directories

- **Syntax**

  cp <source file> <new file name>

- **Example**

  cp file1 file2

Command - cp

- **Options to cp**

-p

Copies the file and preserves the following attributes

- owner id
- group id
- permissions
- last modification time

-r

- recursive copy; copy subdirectories under the directory if any

-i

- interactive; prompts for confirmation before overwriting the target file, if it already exists

Command - mv

- **mv** command is used to move a file, or rename a file

- Preserves the following details
  - owner id
  - group id
  - permissions
  - Last modification time
- **Options**

  -f suppresses all prompting (forces overwriting of target)

  -i prompts before overwriting destination file

Command - rm

- The **rm** command is used to remove a file

- **Syntax :**

  rm file(s)

- **Options**

  -f suppresses all prompting

  -i prompts before deleting destination file

  -r will recursively remove the file from a directory (can be used to delete a directory along with the content)

- **Caution:** Use “i” option along with “r” to get notified on deletion

Hard Link & Soft Link

- **Linking files**
  - **Hard Link (in the same file system)**

    $ ln /usr/bin/clear /usr/bin/cls
    - **Note:** Hard links uses the same inode number
  - **Soft Link (possible across file systems - also used to link directories)**

    $ ln –s /usr/bin/clear /home/user1/cls

find

- Lets user to search set of files and directories based on various criteria

- **Syntax:**

  find [path...] [expression]

  [path]
  - where to search

    [expression]
  - What type of file to search (specified with –type option)
  - What action to be applied (–exec, –print, etc.)
  - Name of the files (specified as part of –name option, enclosed in “ “)
- **Example**

  find . –name “*.c” -print

  lists all files with .c extension from the current dir & its subdirectories

find (Contd.).

- **Finding files on the basis of file size**
  - size [+ –]n[bc]

  n represents size in bytes (c) or blocks (b) of 512 bytes
- **Examples:**

  find . –size 1000c => lists all files that are exactly 1000 bytes in size

  find . –size +1000c => lists all files that are more than 1000 bytes in size

  find . –size –1000c => lists all files that are less than 1000 bytes in size

find (Contd.).

- **Finding files on the basis of access time (atime) or modified time (mtime)**
  - atime [+-]n
  - mtime [+-]n

  n represents number of days (actually 24 * n hours)

- **Examples:**

  find . –atime 2 => lists files accessed exactly 2 days ago

  find . –atime +2 => lists files accessed more than 2 days ago

  find / –mtime –2 =>lists files modified less than 2 days ago

find (Contd.).

- **Few more options with find:**

-inum n => find files having the inode number n

- type filetype => find files of the file type

f - for ordinary files

d – for directories

-newer fname => find files newer than fname

-perm mode => find files of the specified permissions

-user username => find files owned by the username

find (Contd.).

- **Applying a command on files matching the criteria with – exec and –ok options**

– exec command {} \;

- command is to be applied on the matching files (does not prompt user) – Non interactive.

  find . -name "*.dat" –exec ls –l {} \;

  The above command will long list all files with .dat extension in the current and its subdirectories.

-ok command {} \;

- Functionality is similar to –exec, but prompts user before applying the command on the file matching the criteria.

grep

- **grep –Global Regular Expression Printer is used for searching regular expressions**

- **Syntax**

  grep <options> <pattern> <filename(s)>

- **Related commands**

  egrep & fgrep

grep options

-c displays count of the number of lines where the pattern occurs

-n displays line numbers along with the lines

-v displays all lines except lines matching pattern

-i Ignores case for matching

```
find . -name "*.dat" –exec ls –l {} \;| grep -c back
find . -name "*.dat" –exec ls –l {} \;| grep -v back
```

sed: **sed** command is mostly used to replace the text in a file.

- sed 'G' /etc/passwd
- sed '4G' /etc/passwd
- sed '10G' /etc/passwd
- sed '4!G' /etc/passwd
- sed '4,6!G' /etc/passwd

- Assigns gap or gives no gap

- ==========

- sed 's_/sbin/nologin_/bin/all_'< /etc/passwd
- sed '/sshd/s_/sbin/nologin_/bin/bash_' < /etc/passwd
- sed '/sshd/!s_/sbin/nologin_/bin/bash_' < /etc/passwd
-
-

```
  141  sed '/root/p' /etc/passwd
  142  sed '/root/p' /etc/passwd | less
  143  sed -n '/root/p' /etc/passwd | less
  144  sed '/root/p' /etc/passwd | less
  145  sed -n '1,4w /tmp/output_file' /etc/passwd
  146  ls /tmp
  147  cat /tmp/output_file
==============================
  148  cp /etc/passwd ./passwd
 ==============================
  159  sed '2d' ./passwd  {deletes 2nd Line }
  161  sed '1,10d' ./passwd  | less
  162  sed '1,10!d' ./passwd  | less
==============================
  164  sed 's/^[\t]*/*hello*/' ./passwd  {inserts word *hello* }
  165  sed 's/^[\t]*/<>/' ./passwd
  166  sed 's/^[\t]*/#/' ./passwd
==============================
 176  sed 's/^[\t]*/\n/' ./passwd {inserts gap }
 181  sed '/^[\t]*/G' ./passwd | less
```

File system commands

- **df - To check the Filesystem size**
  - **Syntax:**

    df [-kmih]

    $ df -k

    Filesystem 1k-blocks Used Available Use% Mounted on
    /dev/hda1 8064272 6339628 1314992 83% /
    /dev/hda3 4032161 2016080 1016081 50% /home

- **du - To check file space usage**
  - **Syntax:**

    du [-ks]

    $du –s

    83504 .

The awk Programming Language

- **awk** is to give Linux a general purpose programming language that handles text (strings) as easily as numbers.
- nawk (new awk) is the new standard for Awk - designed to facilitate large awk programs.
- Looks at data by records and fields
- awk can take the input from
  - files
  - redirection and pipes
  - directly from standard input
- Specify individual records with $0
- Specify individual fields with $1, $2, $3, and so on

awk - examples

- awk -F: '{print $1, $3}' /etc/passwd

Prints the first and 3rd fields(User name and user id) from the file /etc/passwd.

- awk -F: '$3>99{print $1, $3}' /etc/passwd

Prints the first and 3rd fields(User name and user id) from the file /etc/passwd only if the user id is greater than 99.

- awk-F: 'NR==1,NR==3{print $0}' /etc/passwd

Prints the first 3 rows from the file /etc/passwd

- who | awk '{print $1}' | uniq

print the unique list of users who are currently logged in.

- awk 'END{print NR}' /etc/passwd

print the number of lines in the file /etc/passwd

awk

```
 1. Field Processor
 2. Supports grep-style (POSIX) regular expressions
 3. Default field-delimiter is whitespace
 4. Stores fields (columns) into tokens, which then become accessible during processing
 5. Loops over input one line at a time
 6. Will accept input from: file or STDIN or pipe

Tasks:
 1. awk '{ print $0 }' grep.test.txt - prints each line in its entirety
 2. awk '{ print $1 }' grep.test.txt - prints column #1 from each line
 3. awk '{ print $2 }' grep.test.txt - prints column #2 from each line
 4. awk '{ print $2,$1 }' grep.test.txt - prints column #1 then #2
 5. awk '/Red/ { print $0}' grep.test.txt - prints ALL columns where line includes 'Red'
 6. awk '/Red/ { print $1,"-",$2,"-",$3}' grep.test.txt - prints ALL columns, with
transformations, where line includes 'Red'
 7. awk '{ if ($2 ~ /2011/) print $0 }' grep.test.txt - prints ALL columns of records
containing '2011' in the second column
 8. awk '/2011$/ { print $0 }' grep.test.txt - prints lines ending in: '2011'
 9. awk '/2011$/ { print $0 }' - waits on STDIN for input
10. grep 2011 /var/log/messages | awk '/2011$/ { print $0 }' - accepts a pipe
11. awk '{ if ($2 ~ /9/) print $3,$4,$5,$6 }' /var/log/messages - prints columns $3-$6 where
colum 2 = '9'
12. awk 'NR %2 == 0' <filename> -- prints even lines in a file
13. awk 'NR %2 == 1' <filename> -- prints odd lines in a file
```

Run Levels

- Each of the run levels of Linux is a different "operating mode"

- **single user mode:**
  - checking or backing up of file systems done
  - only one user – root
- **multiuser mode:**
  - All the file systems are mounted
  - System services (daemons) are started

/etc/inittab

- When you boot the system or change run levels with the init or shutdown command, the init daemon starts processes by reading information from the /etc/inittab file.

- This file defines the following important items for the init process:
  - That the init process will restart
  - What processes to start, monitor, and restart if they terminate
  - What actions to take when the system enters a new run level

Changing the run level

- **who –r or runlevel**

  command to display the current run level of the system

- To switch between run levels you use the init command:

  #init <Run Level Number>

- **Examples:**

  init 1 => switch to single user mode

  init 6 => reboot the system

Summary

In this module, you have learned about:

- Introduction to bash shell
- Getting Started - Shell prompts, Linux Command - arguments & options
- Printing the working directory - pwd
- Getting help on commands
- Managing files & directories
- Hard link & Soft Link
- find command - to search for files and directories
- Filters - fgrep, grep & egrep
- File system commands – df & du
- The awk Programming Language
- Working with run levels

Review Questions

1. You can not set the permission of a directory when it's getting created.
   a. TRUE
   b. FALSE

2. The command **find** is used to search for a specific text/ pattern in file.
   a. TRUE
   b. FALSE

Hands-on Exercises

EduRamp ELs
The B2B Learning Specialist

References

- Das, Sumitabha. *Linux : Concepts and Applications*. New Delhi: Tata McGraw-Hill, 2008.
- Eric Steven Raymond.(2003). *The Art of Linux Programming*. Retrieved on August 4, 2011, from, http://www.faqs.org/docs/artu/ch17s02.html
- Bob Rankin. *What is a Linux Virtual Console?*. Retrieved on September 17,2011, from, http://lowfatlinux.com/linux-virtual-console.html#ixzz1Zygire6V
- Wikimedia Foundation, Inc.(2011). *Linux*. Retrieved on September 18, 2011,from, http://en.wikipedia.org/wiki/Linux_System_V
- Wikimedia Foundation, Inc.(2011). *Linux Architecture*. Retrieved on September 18, 2011, from, http://en.wikipedia.org/wiki/Linux_architecture

Objectives

- At the end of this module you will learn about:
- System Administration
- Linux Networking
- Linux Native Performance Tools

System Administration

EduRamp ELs
The B2B Learning Specialist

Objectives

At the end of this module you will learn about:

- Basic Commands - date, who, id, whoami, who am i, uname, whereis, tty
- vi Editor
- Filters - tee, wc, tr, cut, sort, head, tail, more, less
- The file command
- Shutting the system down
- System Directories

Basic Commands

- **date**
  - This command displays the current system date and time on the screen.

Basic Commands (Contd.).

- **who**
  - Displays the names of all the users who have currently logged in
- **id**
  - Displays your UID, Primary GID, and Secondary GID's
- **whoami**
  - Displays the effective user ID
- **who am i**
  - Displays the name of the current user

Basic Commands (Contd.).

- **uname**
  - The uname utility prints information about the current system on the standard output.

- **whereis**
  - Displays the path/ location of a command

- **tty**
  - Prints the terminal's name

vi editor

- **vi** is a visual editor used to create and edit text files.
  - A screen-oriented text editor
  - Included with most Linux system distributions
  - Command driven
- Categories of commands include
  - Cursor movement
  - Editing commands
  - Search and replace commands
- The vi editor is invoked by the following command:
  - $ vi filename

vi modes – the three modes of vi

- **vi** functions in the following modes:
  - Command Mode
  - Insert Mode
  - Last Line Mode

vi command mode

- In the command mode, you can do the following:
  - Issue commands to insert, append, delete, copy etc.
  - Navigate the file
  - Search for text

vi insert mode

- The vi editor switches to insert mode when you issue insert, append or open commands.
- In this mode, you can type text into your file and use the arrow keys to navigate around your file.
- To switch from insert mode to command mode, you need to press the [Esc] key.

vi last line mode

- You can go to the last line mode only from the command mode
- You type **:** in the command mode to switch to the last line mode
- You can do lot of useful actions like saving the file, search and replace commands, issue vi configuration commands.
- You can also issue Linux commands from the last line mode

  :! ls

vi configuration commands

- You run the vi configuration commands in the last line mode.

Find below some useful ones:

:se[t] nu => display the line numbers

: se[t] nonu => remove the line numbers

: se[t] showmode=> display the current vi mode

Navigation

Editing Commands

- **Text insertion / replacement commands**

i - inserts text to the left of the cursor

a - inserts text to the right of the cursor

I - inserts text at the beginning of the line

A - appends text at end of the line

o - opens line below

O - opens line above

R - replaces text from cursor to right

s - replaces a single character with any number of characters

S - replaces entire line

Editing Commands (Contd.).

- **Commands for deletion**

x - to delete character at cursor position

3x - to delete 3 characters at cursor position

dw - to delete word

2dw - to delete 2 word

dd - to delete a line

2dd - to delete 2 lines

cw - to change the text

Editing Commands (Contd.).

- **Yanking (Copying)**

  Y - copy line into buffer
  3Y - copy 3 lines into buffer
  yw - copy a word
  p - paste buffer below cursor
  P - paste buffer above cursor
  u - undo

- **Save and quit**

  :w - to save
  :w! - to name a file (:w! filename -> save as)
  :x - save and quit
  :q - cancel changes
  :q! - cancel and quit

Search & Replace Commands

- **The following commands are applicable for vi editor in Linux:**

/pat searches for the pattern pat and places cursor where pattern occurs.

/ repeat last search

:%s/old/new/g to change every occurrence in the whole file.

:#,#s/old/new/g
where #,# should be replaced with the numbers of the two lines (say between line no.'s 2 and 5).

- **Example -** :2,5s/am/was/g

Filter Command – tee

- tee command allows the normal output to the standard output, as well as to a file
- Useful to capture intermediate output of a long command pipeline for further processing, or debugging purpose
- **Examples**
  - who | tee userlist
  - cat - | tee file1 | wc -l

WC

- **wc**

  A filter used to count the number of lines, words, and characters in a disk file or from the standard input

  -l - displays the number of lines

  -w - displays the number of words

  -c - displays the number of characters

tr command

- **tr - translate filter used to translate a given set of characters**

- **Example :**

  tr [a-z] [A-Z] < filename => This converts standard input read from lower case to upper case

  cat lcasefile | tr “[a-z]” “[A-Z]” >ucasefile

tr (Contd.).

- **Useful options for tr**

-s char

Squeeze multiple contiguous occurrences of the character into single char

-d char

Remove the character

Filter Command – cut

- **The cut command is used to extract specified columns/ characters of a text**

Option	remark
-c	used to extract characters
-d	Delimiter for fields
-f	Field no.

- **Examples:**

  $ cut -c2-5 file1
  $ cut -d "|" -f2,3 file1

- #cut –d ":" –f1,5,6 < /etc/passwd

sort

- **Sorts the contents of the given file based on the first char of each line.**

- **Options**

  -n numeric sort (comparison made according to strings numeric value)
  -r reverse sort
  -t specify delimiter for fields
  +num specify sorting field numbers
  -knum specify sorting filed numbers

- **Examples**

  $ sort +2 filename => will sort according to the 3rd field.
  $ sort –k3 filename => will sort according to the 3rd field.

head

- **The head command will display the top 10 lines by default.**

$ head -3 file1

Displays the first 3 lines of the file

tail

- **The tail command displays the last n lines of a file**

$ tail -3 file1 => Displays the last 3 lines of the file

Can also specify the line number from which the data has to be displayed till the end of file

$ tail +5 file1

Note: The + option is not supported by some Linux/linux versions.

Commands – more, less, file

- **more filename** - Displays the file one page at a time
- **less filename** - Displays the file one page at a time
- **file filename** - Display the details of the file type

Shutting the System down

- **shutdown usually perform the following activities:**
  - notifies users with “wall” command about the system going down
  - send signals to all running processes so that they terminate normally
  - logs users off and kills remaining processes
  - unmounts all the secondary file systems
  - write information about file system status to disk to preserve the integrity of the file system
  - notifies the users to reboot or switch off or moves the system to single user mode

shutdown

- **Examples**

shutdown - system is moved to single user mode

shutdown –g2 - grace period of 2 minutes before shutdown

shutdown –y - don't prompt the admin "do you want to shutdown"

shutdown –y –g0 - do an immediate shutdown

shutdown –y –g0 –i6 - shutdown and reboot

shutdown –y "system going down for backup"
- customized message broadcast

Summary

At the end of this module you will learn about:

- Basic Commands - date, who, id, whoami, who am i, uname, whereis, tty
- vi Editor
- Filters - tee, wc, tr, cut, sort, head, tail, more, less
- The file command
- Shutting the system down
- System Directories

Review Questions

1. The ---------- command is used to display the effective user id along with the effective group and other details.
   a. who
   b. whoami
   c. id
   d. who am i

2. Which of the following system directory stores the manual documents?.
   e. /etc
   f. /var
   g. /usr
   h. /lib

Hands-on Exercises

EduRamp ELs
The B2B Learning Specialist

Hands-on Exercises (Contd.).

Linux Networking

Objectives

- At the end of this module you will learn about:
  - Services / Ports / ping / telnet / ssh / NFS / NIS / DNS / dhcp / netstat
  - Configuring Virtual IP and working with virtual IP

Network Services

- Each network service uses a port that represents an address space reserved for that service.
- A client often communicates with a server through a well-known port.
- Well-known ports are stored in the /etc/inet/services file.

Starting & Stopping Services

- **Using the service command**

  # service httpd start

  # service httpd stop

  # service httpd restart

- **Using the shell scripts in /etc/init.d**

  # /etc/init.d/httpd start

  # /etc/init.d/httpd stop

Network Ports

- Network ports help transport protocols distinguish between multiple service requests arriving at a given host computer.
- The TCP and UDP transport protocols identify ports using a positive integer between 1 and 65535, which is called a port number.
- Network ports can be divided into two categories, well-known ports and ephemeral (short-lived) ports.

The ping command

- Ping is a network administration utility used to test the reachability of a host on a network .
- It measures the round-trip time for messages sent from the originating host to a destination computer.
- Ping operates by sending Internet Control Message Protocol(ICMP) echo request packets to the target host and waiting for an ICMP response.
- In the process it measures the time from transmission to reception (round-trip time) and records any packet loss.

The ssh command

- ssh uses public-key cryptography to authenticate the remote computer and allow the remote computer to authenticate the user, if necessary.
- ssh is typically used to log into a remote machine and execute commands,.
- **syntax:**

  ssh [-l loginname] hostname | user@hostname [command]

  eg: ssh user1@10.201.120.13 => connect to 10.201.120.13 as user1

  ssh 10.201.120.13 => connect to 10.201.120.13 as the currently logged in user.

  ssh 10.201.120.13 who => executes the who command at 10.201.120.13 as the currently logged in user.

Network File System (NFS)

- The NFS service enables computers of different architectures running different operating systems to share file systems across a network.
- Allows multiple computers to use the same files, since all the users on the network are able to access the same data.
- Storage costs is reduced by sharing applications on computers instead of allocating local disk space for each user application
- Provides data consistency and reliability, because all users can read the same set of files

Network Information Service(NIS)

- NIS facilitates the creation of server systems that act as central repositories for several of the administrative files found on Linux systems.

- **The benefits of NIS include:**
  - Centralized administration of configuration files
  - Better scaling of configuration file administration as networks grow

Domain Naming System(DNS)

- Domain Name System (DNS) is an Internet-wide naming system for resolving host names to IP addresses and IP addresses to host names.

- DNS supports name resolution for both local and remote hosts, and uses the concept of domains to allow hosts with the same name to coexist on the Internet, so long as they are in different domains..

Dynamic Host Configuration Protocol (dhcp)

- DHCP enables you to provide network-related information to client systems through a centrally located server system.
- DHCP evolved from the bootstrap protocol (BOOTP).
- DHCP reduces the cost of managing networks by eliminating the need to manually assign or change IP addresses repeatedly.
- DHCP also reclaims IP addresses that are no longer needed or if the time period for their use has expired. These IP addresses can then be used by other clients.

netstat

- The command **netstat** displays network connections, routing tables, interface statistics, masquerade connections, and multicast memberships

- **Example :**

  command to display the Interface statistics

  $ netstat –i

  command to display routing table

  $ netstat –nr

The ifconfig command

- Ifconfig is a system administration utility in Linux-like operating systems to configure, control, and query TCP/IP network Interface parameters.

- **Common uses for ifconfig include:**
  - setting an interface's IP Address and netmask,
  - disabling or enabling a given interface

Examples

ifconfig eth0 up => to activate the network interface eth0

ifconfig eth0 down => to bring down the network interface eth0

Virtual IP Address

- virtual IP address (VIP or VIPA) is an IP address that is not connected to a specific computer or network interface card (NIC) on a computer. Incoming packets are sent to the VIP address, but they are redirected to physical network interfaces.
- VIPs are mostly used for connection redundancy; a VIP address may still be available if a computer or NIC fails, because an alternative computer or NIC replies to connections.
- A virtual IP address eliminates a host's dependency upon individual network interfaces.

Advantages of Virtual IP Address

- Advantages of Virtual IP Addresses

  In spite of their simplicity, virtual IP addresses offer two main advantages over their physical counterparts:
  - High availability
  - Unlimited mobility

- These advantages mainly stem from the fact that virtual IP addresses are purely virtual and are not bound to a physical network wire. These two unique advantages are explored below in details.

Configuring Virtual IP Address using linuxconf

- Invoke the linuxconf tool. (type linuxconf in the shell prompt and press Enter). For starting this tool you have to be logged in as Super user.
- In the displayed menu, select the menu item : IP Aliases for virtual hosts and press Enter. This will display the two interfaces eth0 and lo. (Ethernet interfaces in Linux are called by such names as eth0 and eth1)
- Select eth0 and press Enter. This will show all the IP aliases configured for the selected interface.
- In the field IP alias or range, enter the desired IP address or the range with a corresponding Netmask.

  Example

  IP alias 177.177.177.15 Netmask : 255.255.255.0

Configuring Virtual IPs in the Linux box at run-time

- Using the following command, we can configure Virtual IPs in the Linux box at run-time without re-booting the machine or re-starting the network.

```
ifconfig  eth0:1  172.19.1.5  netmask  255.255.255.0  up
ifconfig  eth0:3  172.19.1.6  netmask  255.255.255.0  up
```

- The same interfaces can be made down by using the following command.

```
ifconfig  eth0:1  172.19.1.5  netmask  255.255.255.0  down
ifconfig  eth0:3  172.19.1.6  netmask  255.255.255.0  down
```

Summary

In this module, you have learned about:

- Services / Ports / ping / telnet / ssh / NFS / NIS / DNS / dhcp / netstat
- Configuring Virtual IP and working with virtual IP

Review Questions

1. The ---------- helps transport protocols distinguish between multiple service requests arriving at a given host computer.
   a. IP Address
   b. Host Name
   c. Network Port
   d. None

2. The **NIS** service allows the administrator to set the IP Address dynamically.
   a. TRUE
   b. FALSE

Linux Native Performance Tools

EduRamp ELs
The B2B Learning Specialist

At the end of this module you will learn about:

- top/topas
- iostat / vmstat / sar

List of Commands and Utilities

- **top / topas** – Real time process navigation
- **iostat** – Input output statics
- **vmstat** – memory and virtual memory statistics
- **sar -** cpu, queuing, paging, file access, and more

top / topas (AIX) – real-time process status

The top command displays currently running processes and important information about them including their memory and CPU usage. The list is both real-time and interactive.

top

```
top - 15:02:46 up 35 min, 4 users, load average: 0.17, 0.65, 1.00
Tasks: 110 total, 1 running, 107 sleeping, 0 stopped, 2 zombie
Cpu(s): 41.1% us, 2.0% sy, 0.0% ni, 56.6% id, 0.0% wa, 0.3% hi, 0.0% si
Mem: 775024k total, 772028k used, 2996k free, 68468k buffers
Swap: 1048568k total, 176k used, 1048392k free, 441172k cached

PID  USER    PR NI  VIRT  RES  SHR  S %CPU %MEM TIME+   COMMAND
4624 root    15  0  40192 18m  7228 S 28.4  2.4  1:23.21  X
4926 mhideo15  0  55564  33m 9784 S 13.5  4.4  0:25.96  gnome-terminal
6475 mhideo16  0   3612  968  760  R  0.7  0.1  0:00.11  top
```

IOSTAT / VMSTAT

- The **iostat** utility displays kernel I/O statistics on terminal, device and cpu operations.
- **Example :**

  $ iostat –t 10 => Display tty statistics every 10 seconds

- **vmstat** reports information about processes, memory, paging, block IO, traps, and cpu activity.
- **Example :**

  $ vmstat 2 5 => To display five summaries at 2-second intervals .

sar

- The sar command is mainly used for gathering statistical information about your system - cpu, queuing, paging, file access, and more — that can help determine system performance.

- The sar command can be used for:
  - Collecting real-time information
  - Displaying previously captured data
  - Collecting data using cron

- **Example**

$ sar => Displays todays CPU activity so far.

Summary

- In this module, you have learned about:
  - top/topas
  - iostat / vmstat / sar

Review Questions

1. All the users can run the command **top**.
   a. TRUE
   b. FALSE

2. The output of the command **iostat** will be always same.
   a. TRUE
   b. FALSE

References

- Das, Sumitabha. *Linux : Concepts and Applications*. New Delhi: Tata McGraw-Hill, 2008.
- Paul K. Andersen. Just Enough Linux. New York: McGraw-Hill, Inc, 2006.
- Stan, Peter Klimas. (2009). Linux Newbie Administrator Guide. Retrieved on September 16, 2011,from, http://lnag.sourceforge.net/downloads/LinuxNewbieAdministratorGuide.pdf
- TechguruLive (2010). *How to configure Virtual IP Address in Linux.* Retrieved on October 4, 2011, from, http://techgurulive.com/2009/03/07/how-to-configure-virtual-ip-address-in-linux

References (Contd.).

- Olaf Kirch.(1993),Terry Dawson(2000).Linux network Administrators Guide. Retrieved on September 30,2011,from, http://tldp.org/LDP/nag2/nag2.pdf

- Webnms.com(2010). Configuring IP addresses. Retrieved on November 9, 2012, from, http://www.webnms.com/simulator/help/sim_network/netsim_conf_virtual_ip.html

- Novel.com(2003). Virtual IP Addresses in the NetWare 6.5 TCP/IP Stack. Retrieved on November 9, 2012, from, http://support.novell.com/techcenter/articles/ana20030803.html

References (Contd.).

- Novel.com(2012). Virtual IP Address Benefits. Retrieved on November 9, 2012, from, http://www.novell.com/documentation/bcc/bcc11_admin_nw/data/bq7ucwl.html

- Wiki.archlinux.org(2012). Secure Shell Retrieved on November 9,2012,from, https://wiki.archlinux.org/index.php/Secure_Shell

Objective

At the end of this module you will learn about:

- User Management
- Process Management
- Job Scheduling and Feature Differences between Flavors of Linux
- Linux Backup Tools

Objectives

At the end of this module you will learn about:

- Understanding File and Directory Permissions
- chown & chgrp
- umask command
- Creating, modifying and deleting User Accounts
- Creating, modifying and deleting Group Accounts
- The su command – switch between users

User Management

File Access Permissions

- Refers to the permissions associated with a file with respect to the following:

- **Permission Levels**
  - User (owner) (u)
  - Group (wheel, staff, daemon, etc.) (g)
  - World (guest, anonymous and all other users) (o)

- **Permission Settings**
  - Read (r)
  - Write (w)
  - Execute (x)

File Access Permissions (Contd.).

- No read permission does not allow the user to:
  - List the contents of directory
  - Remove the directory

- No Write permission prevent the user to :
  - Copy files to the directory
  - Remove files from the directory
  - Rename files in the directory
  - Make a subdirectory
  - Remove a subdirectory from the directory
  - Move files to, and from the directory

File Access Permissions (Contd.).

- No execute permission does not allow the user to:
  - display the contents of a directory file from within the directory
  - change to the directory
  - display a file in the directory
  - copy a file to, or from the directory

Changing Permissions - chmod

- **Syntax:**

  chmod <category> <operation> <permission> <filename(s)>

  or

  chmod <octal number> filename

- **Octal Number**
  - 4 - for read
  - 2 - for write
  - 1 - for execution

- **Examples**

  $ chmod 744 xyz => This sets read, write and execute permissions for owner, read permission for group and others

  $ chmod u+x file1 => This will add x privilege to the owner.

Command – chown & chgrp

- **chown** – This command changes the owner of the file.

  eg. chown user2 file1 - the new owner of the file file1 is user2.

- **Note:** **This command can be run by the super user(root) only**

- **chgrp** – This command is used to change the group owner of the file.

  eg. chgrp group2 file1 – this will set the group owner of the file file1 as group2.

- **Note:** **This command can be run by a normal user to change the group owner to any of the groups to which he belongs to. If the group has to be changed to some other group, to which he do not belongs to, only superuser can change it.**

umask command

- umask value is used to set the default permission of a file and directory while creating
- **umask** command (without any arguments) is used to see the default mask for the file permission
- Default umask value will be set in the system environment file like/ etc/ profile
- The command **umask 022** will set a mask of 022 for the current session
  - If you create a new file now, the file permission will be 644
  - And the directory permission will be 755

User Management

- Users refer to either people which means accounts tied to physical users or accounts which exist for specific applications to use.
- Groups are used to club users together for a common purpose. Users within a group can generally read, write, or execute files owned by that group.
- Each user and group has a unique numerical identification number called a userid (UID) and a groupid(GID), respectively.
- A user who creates a file is also the owner and group owner of that file.

User Management (Contd.).

- System Administrator uses commands or GUI tool to create User credentials (ID and password)
- **Commands to create a user**
  useradd or adduser
- **Commands to create a group**
  groupadd or addgroup
- **Commands to modify a user & group**
  usermod groupmod
- **Commands to delete a user & group**
  userdel groupdel
- **Command to set/reset password**
  passwd <username>

Creating user account – useradd command

- **Syntax:**

useradd [-c comment] [-d home_dir] [-e expire_date] [-f inactive_time] [-g initial_group] [-G group[,...]] [-m] [-k skeleton_dir] [-p passwd] [-s shell] [-u uid [-o] login

- **Examples**

useradd User1 => command to create a locked user account
User1

passwd User1 => Sets the password for the new user

- **Note:** The above commands make entries in the associated file /etc/passwd & /etc/shadow.

#useradd –D => will display the default values used by the useradd command.

Modifying use account – usermod command

- **Syntax:**

  usermod [-c comment] [-d home_dir [-m]] [-e expire_date] [-f inactive_time] [-g initial_group] [-G group[,...]] [-l login_name] [-p passwd] [-s shell] [-u uid [-o]] [-L|-U] login

- **Note: Most** of the options associated with the useradd command are available with the usermod command as well.

Deleting user accounts – userdel

- **Syntax:**
  userdel [-r] login

-r Files in the user's home directory will be removed along with the home directory itself and the user's mail spool.

- **Note: Files** located in other file systems will have to be searched for and removed manually.

Without the option –r, the command userdel only deletes the user. His home directory will remain intact.

Adding Group Account – groupadd command

- **Syntax:**

  groupadd [-g gid [-o]] group

  -g gid => specifies the Group ID of the group

  # groupadd Group1 => create a new group Group1

Modifying Group Accounts – groupmod command

- **Syntax**

  groupmod [-g gid [-o]] [-n group_name] group

  -n group_name => specifies the new group name. This option is used for renaming a Group Account

Deleting Group Account - groupdel

- **Syntax**

  groupdel groupname

- **Example:**

  # groupdel Group1

User Management – GUI Tool

- In RHEL5 , you can use the GUI Tool User Manager to manage the users & groups in a GUI environment.

- To use the User Manager, you must be running the X Window System, have root privileges, and have the system-config-users RPM package installed.

- To start the User Manager from the desktop,
  - go to System (on the panel) -> Administration -> Users & Groups.
  - you can also type the command system-config-users at a shell prompt (for example, in an XTerm or a GNOME terminal).

passwd

- **To change a user password**

  $ passwd

  Changing password for username

  New Linux password:

  Reenter Linux password:

User Related Files

- **/etc/passwd** This file stores all the user information except the password
- **/etc/shadow** This file stores the encrypted passwords and all password related information
- **/etc/group** This file stores all the groups in the system along with the secondary group members
- **/etc/default/useradd** This files stores the default values used by useradd program
- **/etc/login.defs** This file defines the site specific configuration for the shadow password suite.

/etc/passwd

- **Have a look at the example given below of /etc/passwd file from a Linux system**

root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/bin/bash

Find below the columns of the file:

LoginID - user's login name
x - represents a placeholder for the user's encrypted password
UID - user ID number used by the system to identify the user
GID - GID number which identifies the user's primary group
comment - comment about the user – normally a user's full name
home_directory - specifies the full path name to the user's home directory
login_shell - defines the user's login shell

/etc/shadow

- **Find below the first two lines of an /etc/shadow file from a linux system:**

  root:5RiJS.yvdGBkU:13255:0:99999:7:::
  bin:*:13255:0:99999:7:::

Fields in the /etc/shadow file

loginID - user's login name
encrypted_passwd - encrypted password
lastchg - the number of days between Jan 1, 1970 and the last password modification date
min - minimum no. of days between password changes
max - maximum password age (number of days the password is valid)
warn - The 'warn' field flashes the information about the number of days before the password expires
inactive - number of inactive days allowed for the user before the account getting locked
expire - The date when the user account expires
reserved field - For future use

/etc/group

- **Find below the first two lines of an /etc/group file from a linux system**

  root:x:0:root

  bin:x:1:root,bin,daemon

- **Fields in the /etc/group file**

goupname - name assigned to the group

group-passwd - allows a non group member to work as group member on supply of this password

gid - group id

userlist - list of user names for whom this group is their secondary group.

/etc/default/useradd

- **Find below content of /etc/default/useradd file from a linux system**

```
# useradd defaults file
GROUP=100
HOME=/home
INACTIVE=-1
EXPIRE=
SHELL=/bin/bash
SKEL=/etc/skel
CREATE_MAIL_SPOOL=yes
```

/etc/login.defs

- **Find below few lines of /etc/login.defs file from a linux system**

```
MAIL_DIR                /var/spool/mail
PASS_MAX_DAYS           99999
PASS_MIN_DAYS           0
PASS_MIN_LEN            5
PASS_WARN_AGE           7
UID_MIN                 500
GID_MIN                 500
CREATE_HOME             yes
```

su command

- The su command is used to switch the user to another user
- System administrator should not login as “root” . They should login as a normal user and run su command to switch user
- **Examples**

  su => Profile will not change

  su – => Profile will change to the root’s

  su - <username> => Switch to a user with profile change

Summary

In this module, you have learned about:

- Understanding File and Directory Permissions
- chown & chgrp
- umask command
- Creating, modifying and deleting User Accounts
- Creating, modifying and deleting Group Accounts
- The su command – switch between users

Review Questions

1. Any user can execute the command **chown** to change the ownership of his files.
   a. TRUE
   b. FALSE

2. The **su** command without any arguments will switch the user back to the login user.
   a. TRUE
   b. FALSE

Hands-on Exercises

EduRamp ELs
The B2B Learning Specialist

Process Management

EduRamp ELs
The B2B Learning Specialist

Objectives

- At the end of this module you will learn about:
  - Discussion about pipes and sockets
  - Process, Process Status, Foreground and Background Processes
  - Killing Processes
  - Process Priority
  - Accessing system and boot security logs

Pipes and Sockets

- Pipes and Sockets are special files that programs use to communicate with one another.

- **Unnamed pipe:**

  A simple example of using an unnamed pipe is the command:

  ls | grep x => bash and other shells run both commands, connecting the output of the first to the input of the second.

- **Named pipe:**

  Instead of a conventional, unnamed, shell pipeline, a named pipeline makes use of the file system. It is explicitly created using mkfifo() or mknod() and two separate processes can access the pipe by name — one process can open it as a reader, and the other as a writer.

- **Example: Execute in Virtual Console1** **Execute in Virtual Console2**

  #mkfifo pipe1 ; ls –l > pipe1 # cat < pipe1

  The output of the first command shows up on the second console.

Named pipe - Examples

- **Simple Example:**

Create named pipes

$ mkfifo myfifo ; mkfifo myfifo2

Start a process and redirect standard output stream to the pipe

$ ls -l > myfifo &
[1] 9823

start a process that reads contents from the pipe

$ cat myfifo
insgesamt 0
prw-r--r-- 1 mario users 0 13. Nov 22:37 myfifo
prw-r--r-- 1 mario users 0 13. Nov 22:37 myfifo2

Note : You can create a dump of an Oracle database schema and imports the dump into another remote Oracle database. The size of the database dump does not matter, because the data stream is not stored to hard disks. The data stream will be redirected to the Oracle import tool.

Linux Domain Socket or IPC Socket

- A Linux domain socket or IPC socket is a data communications endpoint for exchanging data between processes executing within the same host operating system.

- While similar in functionality to named pipes Linux domain sockets may be created as byte streams or as datagram sequences, while pipes are byte streams only.

- The programmer's application interface (API) for Linux domain sockets is similar to that of an Internet socket, but does not use an underlying network protocol for communication.

What is a Process

- When program is executed, a new process is created
- The process is alive till the execution of the program is complete
- Each process is identified by a number called pid

Login shell

- As soon as the user logs in, a process is created which executes the login shell.

- Login shell is set for each login in /etc/passwd file.

ps - process status

- The ps command is used to display the characteristics of a process.
- It fetches the pid, tty, time, and the command which has started the process.
- **Options**
  - -f lists the pid of the parent process also.
  - -u lists the processes of a given user
  - -a lists the processes of all the users
  - -e lists all the processes including the system processes
  - -l list the information like process state as well

USER = user owning the process
PID = process ID of the process
%CPU = It is the CPU time used divided by the time the process has been running.
%MEM = ratio of the process's resident set size to the physical memory on the machine
VSZ = virtual memory usage of entire process
RSS = resident set size, the non-swapped physical memory that a task has used
TTY = controlling tty (terminal)
STAT = multi-character process state
START = starting time or date of the process
TIME = cumulative CPU time
COMMAND = command with all its arguments

Foreground & Background Processes

- **Foreground Processes**
  - By default the processes get executed in the foreground.
  - Only one process can get executed in the foreground at a time in one terminal.

- **Background Processes**
  - If the command terminates with an ampersand (&), Linux executes the command in the background
  - Background processes enables the user to do more than one task at a time.
  - Shell returns by displaying the process ID (PID) and job id of the process

Controlling Background Process

- **jobs**

  List the background process

- **fg %<job id>**

  Runs a process in the foreground

- **jobs –l**

The kill Command

- **kill -** Kills or terminates a process
- **kill command send a signal to the process**
  - The default signal is 15 (SIGTERM)
- **kill -9 (SIGKILL)**
  - Terminates the process abruptly
- **pkill : command used to kill a process by name**

  pkill <command name>

  pkill -9 <command name>

nohup Command

- **nohup**
  - Lets processes to continue to run even after logout
  - The output of the command is sent to nohup.out if not redirected

- **Syntax**

  nohup command arguments

- **Example:**

  nohup sort emp.lst &

  [1] 21356

  nohup: appending output to ‘nohup.out’

Process priority

- **nice**
  - This command is used to alter the priority of jobs
- **renice**
  - Used to change the priority of a running process

Accessing Logs

Logs serve several purposes:

- They help us troubleshoot virtually all kinds of system and application problems.
- They provide valuable early warning signs of system abuse.
- When all else fails (whether that means a system crash or a system compromise), logs provide us with crucial forensic data.

Configuring syslog

- The **syslog.conf** file is the main configuration file for the **syslogd** which logs system messages on Linux systems. This file specifies rules for logging.

- Every rule consists of two fields, a selector field and an action field. These two fields are separated by one or more spaces or tabs.

- The selector field specifies a pattern of facilities and priorities belonging to the specified action.

- Lines starting with a hash mark ("#") and empty lines are ignored.

Configuring syslog (Contd.).

- **Some sample syslog entries:**

  *.=crit;kern.none /var/adm/critical => This will store all messages with the priority crit in the file /var/adm/critical, except for any kernel messages.

  kern.crit @Server1 => directs all kernel messages of the priority crit and higher to the remote host Server1.

  kern.* /var/adm/kernel => direct any message that has the kernel facility to the file /var/adm/kernel

  kern.crit /dev/console => directs these messages to the actual console, so the person who works on the machine will get them, too.

- **Note:** After making any kind of changes in the syslog.conf file, you need to restart the syslogd daemon or send the HUP signal to the syslog daemon to get the changes effective.

Important log files

- Most of the log files are stored at the location / var/ log

- Common Linux log files name and usage

/var/log/messages : General message and system related stuff.
/var/log/auth.log :Authentication logs.
/var/log/kern.log : Kernel logs.
/var/log/boot.log : System boot log.
/var/log/secure : Authentication log.
/var/log/utmp : Online users data – used by **who** command
/var/log/wtmp : Login details – used by **last** command

dmesg is a command on most Linux and Linux based operating systems that prints the message buffer of the kernel.

Summary

In this module, you have learned about:

- Discussion about pipes and sockets
- Process, Process Status, Foreground and Background Processes
- Killing Processes
- Process Priority
- Accessing system and boot security logs

Review Questions

1. Only the root user can run the **kill** command and kill the processes.
   a. TRUE
   b. FALSE

2. A user can run multiple processes in the back ground at the same time in a terminal.
   a. TRUE
   b. FALSE

Hands-on Exercises

Job Scheduling & Feature Differences between Flavors of Linux

EduRamp ELs
The B2B Learning Specialist

Objectives

At the end of this module you will learn about:

- Scheduling jobs through at & cron
- Managing access permissions to at & cron utilities
- Feature Differences between Flavors of Linux

at – one time scheduling

- The **at** command is used to schedule commands to be executed once at a particular time in the future.

- **Format of the at command:**

  at –m –q queuename time date
  at –r job
  at –l

  -m => sends mail to the user on completion
  -r job => remove a scheduled job from the queue
  -q queuename => specifies a specific queue
  time => specifies the time of command execution
  -l => lists all jobs scheduled for the user
  date => optional – month name followed by a day number or a day of week

at - Example

```
$ at 15:35
at>echo "hello" > /dev/ttyp2
at> ^D

$ at 10am mar 31 2008
at>echo "taxes due" | mail jones
at> ^D

$ atq
$ atrm  job-id
```

at – Example (Contd.).

```
# at 10:00 am
at> ls  –l  /   >lsout
at> <Control-D>

# at –l          => list the at jobs
1516072300.a

# at –r  1516072300.a        => remove the particular job

# at –r                              => remove all the at jobs scheduled

# ls –l /var/spool/cron/atjobs     => list all at jobs in the queue
```

Controlling Access to the at command

/etc/cron.d/at.deny

- this file exists by default
- list the users who can not schedule at jobs
- referred only when /etc/cron.d/at.allow dont exist

/etc/cron.d/at.allow

- this file dont exist by default
- list the users who can schedule at jobs
- referred first

Interaction between at.allow & at.deny

- If at.allow exists, only users in this file can schedule
- If at.allow do not exist, then only the users whose name is not listed in the at.deny can schedule the atjobs
- If neither file exists, only root can schedule atjobs

- If at.allow exists and is empty, not even root can schedule the at jobs
- If at.allow do not exist, and at.deny is empty then all can schedule the at jobs

Crontab

- **How to add crontab entry**

  $ **crontab –e**

- **Example**

  A line in crontab file like below removes the tmp files from /home/someuser/tmp each day at 6:30 PM.

  30 18 * * * rm /home/someuser/tmp*

- **Possible values for the fields**

n	where n can be within the allowed range for the field
n,p,q	matches the values of n, p and q
n-p	matches within the range – both n & p inclusive
*	matches all allowed values

Crontab (Contd.).

min	hour	day/month	month	day/week	Execution time
30	0	1	1,6,12	*	-- 00:30 Hrs on 1st of Jan, June & Dec.
0	20	*	10	1-5	--8.00 PM every weekday (Mon-Fri) only in Oct.
0	0	1,10,15	*	*	-- midnight on 1st ,10th & 15th of month
5,10	0	10	*	1	-- At 12.05,12.10 every Monday & on 10th of every month

controlling access to the crontab command

/etc/cron.d/cron.deny

- this file exists by default
- list the users who can not schedule crontab jobs
- referred only when /etc/cron.d/at.allow dont exist

/etc/cron.d/cron.allow

- this file dont exist by default
- list the users who can schedule crontab jobs
- if exists, only this file will be referred

[The interaction between cron.allow & cron.deny is exactly similar as that of at.allow & at.deny

Summary

In this module, you have learned about:

- Scheduling jobs through at & cron
- Managing access permissions to at & cron utilities
- Feature Differences between Flavors of Linux

Review Questions

1. The ---------- scheduler is used to schedule a recuring task.
   a. at
   b. batch
   c. crontab
   d. Linux Task Scheduler

2. By default all users are allowed to schedule at jobs.
   a. TRUE
   b. FALSE

Hands-on Exercises

Linux Backup Tools

Objectives

At the end of this module you will learn about:

- Backup through tar /cpio / dd commands
- Recovery single/multiple files

Backup & Restore

- One of the major activities to enable the availability of system and software.
- Each Linux flavor got its own backup functions.
- General backup facility will allow to use the backed up files across flavors.
- Commercial backup facilities are also available.

Type of Backups

- Full Backup
  - Take the backup of all files
- Incremental Backup
  - Take the backup of all files modified after the last full backup or incremental backup
- Differential Backup
  - Take the backup of all files modifies after the last full backup

Backup Utilities: Tape Archive - tar

- **tar** is an archiving utility to store and retrieve files from an archive, known as tarfile.
- Though archives are created on a tape, it is common to have them as disk files as well.
- **Syntax**

  tar c|t|x [vf destination] source...

- **Examples**

  $ tar -cf tar1 emp => take a backup of emp directory into the tarfile tar1

  $ tar –tf tar1 => List the files & directories in the tar file.

Tape Archive - tar

- **Examples:**

  Create a new tar file containing all .dat files (assuming a.dat, b.dat and c.dat exist)

  $ tar –cf mytar *.dat

  $ tar –xf mytar => Restores the data from the backup.

Backup through cpio

- The cpio command is one of the most commonly used Linux back up tools.
- The cpio command has two unusual features. Unlike tar, in which the files to back up are typed in as part of the command, cpio reads the files to work with from the standard input (in other words, the screen).
- This feature means that cpio must be used as part of a multiple command or with a redirection pipe.
- cpio must always be used with one of three flags – extract , create and pass-through.

Backup & restore using cpio - Examples

- **Example: To take the backup of the directory /home/User1**

  # find /home/User1 -print | cpio -ocv > /opt/User1_backup.cpio

- **Command to restore the directory from the backup:**

  # cd /home/User1

  # cpio –icuvd < /opt/User1_backup.cpio

- **Copy a directory structure – using the pass through mode.**

  #find . –depth | cpio –pmdv /export/out

Backup & restore using dd command

- The dd command is used by the Linux kernel Makefiles to make boot images.
- Like most well-behaved commands, dd reads from its standard input and writes to its standard output, which could be altered by a command linen specification.
- This allows dd to be used in pipes, and remotely with the rsh remote shell command.

Backup & restore using dd command - Examples

Full Hard Disk Copy:

```
# dd  if=/dev/hdb  of=/dev/hdc
# dd  if=/dev/hdb  of=/opt/hdb_backup
# dd  if=/dev/hdb | gzip > /opt/hdb_backup.gz
```

Restore Backup of hard disk copy

```
# dd  if=/opt/hdb_backup  of=/dev/hdb
# gzip -dc /opt/hdb_backup.gz | dd of=/dev/hdb
```

MBR backup

In order to backup only the first few bytes containing the MBR and the partition table you can use dd as well.

```
#dd  if=/dev/hda  of=/opt/mbr_backup count=1 bs=512
```

MBR restore

```
dd  if=/opt/mbr_backup  of=/dev/hda
```

- **Note:** Add "count=1 bs=446" to exclude the partition table from being written to disk. You can manually restore the table.

Summary

In this module, you have learned about:

- Backup through tar /cpio / dd commands
- Recovery single/multiple files

Review Questions

1. The product **NetBackup** is from which vendor?
   a. IBM
   b. HP
   c. Veritas
   d. Legato

2. The backup utility **tar** do not come part of the Linux os and has to be separately installed.
   a. TRUE
   b. FALSE

Hands-on Exercises

References

- Das, Sumitabha. *Linux : Concepts and Applications*. New Delhi: Tata McGraw-Hill, 2008.

- Olaf Kirch.(1993),Terry Dawson(2000).Linux network Administrators Guide. Retrieved on September 30,2011,from, http://tldp.org/LDP/nag2/nag2.pdf

- Paul K. Andersen. Just Enough Linux. New York: McGraw-Hill, Inc, 2006.

References (Contd.).

- Ken Milberg.(2009). *Comparing Linux versions: AIX, HP-UX and Solaris*. Retrieved on September 28, 2011, from, http://searchdatacenter.techtarget.com/tip/Comparing-Linux-versions-AIX-HP-UX-and-Solaris

- Stan, Peter Klimas. (2009). *Linux Newbie Administrator Guide*. Retrieved on September 16,2011,from, http://lnag.sourceforge.net/downloads/LinuxNewbieAdministratorGuide.pdf

- Linux-Support.com (2009). *Introduction to Named Pipes and FIFOs*. Retrieved on September 20,2011, from, http://www.linux-support.com/cms/en/component/content/article/1-nuts-and-bolts/65-introduction-to-named-pipes-and-fifos

Thank You