Handout - UNIX And Shell Scripting

Table Of Contents

1. Putty

1. **Does PuTTY support storing settings, so I don't have to change them every time?**

   Yes, all of PuTTY's settings can be saved in named session profiles. You can also change the default settings that are used for new sessions.

2. **Does PuTTY support full-screen mode, like a DOS box?**

   Yes; this is a new feature in version 0.52.

3. **Does PuTTY have the ability to remember my password so I don't have to type it every time?**

   No, it doesn't. Remembering your password is a bad plan for obvious security reasons: anyone who gains access to your machine while you're away from your desk can find out the remembered password, and use it, abuse it or change it.

4. **Where does PuTTY store its data?**

   On Windows, PuTTY stores most of its data (saved sessions, SSH host keys) in the Registry. The precise location is

   HKEY_CURRENT_USER\Software\SimonTatham\PuTTY

   and within that area, saved sessions are stored under Sessions while host keys are stored under SshHostKeys.

5. **How can I start an SSH session straight from the command line?**

   Use the command line putty -ssh host.name. Alternatively, create a saved session that specifies the SSH protocol, and start the saved session

6. **How do I copy and paste between PuTTY and other Windows applications?**

   Copy and paste works similarly to the X Window System. You use the left mouse button to select text in the PuTTY window. The act of selection automatically copies the text to the clipboard: there is no need to press Ctrl-Ins or Ctrl-C or anything else. In fact, pressing Ctrl-C will send a Ctrl-C character to the other end of your connection (just like it does the rest of the time), which may have unpleasant effects. The only thing you need to do, to copy text to the clipboard, is to select it.

   To paste the clipboard contents into a PuTTY window, by default you click the right mouse button. If you have a three-button mouse and are used to X applications, you can configure pasting to be done by the middle button instead, but this is not the default because most Windows users don't have a middle button at all.

7. Who developed putty

   Simon Tatham
8. Putty was written in which language

   C

9. who invented putty

   PuTTY was originally written for Microsoft Windows, but it has been ported to various other operating systems.

10. **PuTTY fails to start up. Windows claims that 'the application configuration is incorrect'.**

    This is caused by a bug in certain versions of Windows XP which is triggered by PuTTY 0.58. This was fixed in 0.59.

11. **Does PuTTY support SSH-1?**

    Yes. SSH-1 support has always been available in PuTTY.

2.Unix Basics

12. How many prompts are available in a UNIX system?
    Two prompts, PS1 (Primary Prompt), PS2 (Secondary Prompt).

13. Name the data structure used to maintain file identification?
    – 'inode', each file has a separate inode and a unique inode number.
    –

14. How does the kernel differentiate device files and ordinary files?
    – Kernel checks 'type' field in the file's inode structure.
    –

15. What is ephemeral port in UNIX?
    – Ephemeral ports are port used by Operating system for client sockets. There is a specific range on which OS can open any port specified by ephemeral port range.

16. How do you find for how many days your Server is up?
    – By using uptime command in UNIX

17. What is Zombie process in UNIX? How do you find Zombie process in UNIX?
    – When a program forks and the child finishes before the parent, the kernel still keeps some of its information about the child in case the parent might need it.

18. How do you set environment variable which will be accessible form sub shell?
    – By using export for example export count=1 will be available on all sub shell.

19. How do you check if a particular process is listening on a particular port on remote host?
    – By using telnet command for example "telnet hostname port", if it suceesfully connects then some process is listening on that port.

20. How do you find whether your system is 32 bit or 64 bit ?
    - Either by using "uname -a" command or by using "arch" command.

3. File Management

21. **How are devices represented in UNIX?**
    All devices are represented by files called special files that are located in /dev directory.

Thus, device files and other files are named and accessed in the same way. A 'regular file' is just an ordinary data file in the disk. A 'block special file' represents a device with characteristics similar to a disk (data transfer in terms of blocks). A 'character special file' represents a device with characteristics similar to a keyboard (data transfer is by stream of bits in sequential order).

22. **What is 'inode'?**
All UNIX files have its description stored in a structure called 'inode'. The inode contains info about the file-size, its location, time of last access, time of last modification, permission and so on. Directories are also represented as files and have an associated inode. In addition to descriptions about the file, the inode contains pointers to the data blocks of the file. If the file is large, inode has indirect pointer to a block of pointers to additional data blocks (this further aggregates for larger files). A block is typically 8k.
Inode consists of the following fields:

- File owner identifier
- File type
- File access permissions
- File access times
- Number of links
- File size
- Location of the file data

23. **Brief about the directory representation in UNIX.**
A Unix directory is a file containing a correspondence between filenames and inodes. A directory is a special file that the kernel maintains. Only kernel modifies directories, but processes can read directories. The contents of a directory are a list of filename and inode number pairs. When new directories are created, kernel makes two entries named '.' (refers to the directory itself) and '..' (refers to parent directory). System call for creating directory is mkdir (pathname, mode).

24. **What are the Unix system calls for I/O?**
1. open(pathname,flag,mode) - open file
2. creat(pathname,mode) - create file
3. close(filedes) - close an open file
4. read(filedes,buffer,bytes) - read data from an open file
5. write(filedes,buffer,bytes) - write data to an open file
6. lseek(filedes,offset,from) - position an open file
7. dup(filedes) - duplicate an existing file descriptor
8. dup2(oldfd,newfd) - duplicate to a desired file descriptor
9. fcntl(filedes,cmd,arg) - change properties of an open file
10. ioctl(filedes,request,arg) - change the behaviour of an open file
11. The difference between fcntl anf ioctl is that the former is intended for any open file, while the latter is for device-specific operations.

25. **How do you change File Access Permissions?**
Every file has following attributes:
- owner's user ID (16 bit integer)
- owner's group ID (16 bit integer)
- File access mode word

(r w x) - (r w x) - (r w x)
(user permission) - (group permission) - (others permission)

To change the access mode, we use chmod(filename,mode).

26. **What are links and symbolic links in UNIX file system?**
A **link** is a second name (not a file) for a file. Links can be used to assign more than one name to a file, but cannot be used to assign a directory more than one name or link filenames on different computers.
Symbolic link 'is' a file that only contains the name of another file.Operation on the symbolic link is directed to the file pointed by the it. Both the limitations of links are eliminated in symbolic links.

Commands for linking files are:
Link "ln filename1 filename2"
Symbolic link "ln -s filename1 filename2"

27. **What is a FIFO?**
FIFO are otherwise called as 'named pipes'. FIFO (first-in-first-out) is a special file which is said to be data transient. Once data is read from named pipe, it cannot be read again. Also, data can be read only in the order written. It is used in inter-process communication where a process writes to one end of the pipe (producer) and the other reads from the other end (consumer).

28. **How do you create special files like named pipes and device files?**
The system call mknod creates special files in the following sequence.
- kernel assigns new inode,
- sets the file type to indicate that the file is a pipe, directory or special file,
- If it is a device file, it makes the other entries like major, minor device numbers.

For example:
If the device is a disk, major device number refers to the disk controller and minor device number is the disk.

29. **Discuss the mount and unmount system calls.**
The privileged mount system call is used to attach a file system to a directory of another file system; the unmount system call detaches a file system. When you mount another file system on to your directory, you are essentially splicing one directory tree onto a branch in another directory tree. The first argument to mount call is the mount point, that is , a directory in the current file naming system. The second argument is the file system to mount to that point. When you insert a cdrom to your unix system's drive, the file system in the cdrom automatically mounts to "/dev/cdrom" in your system.

30. **How does the inode map to data block of a file?**
Inode has 13 block addresses. The first 10 are direct block addresses of the first 10 data blocks in the file. The 11th address points to a one-level index block. The 12th address points to a two-level (double in-direction) index block. The 13th address points to a three-level(triple in-direction)index block. This provides a very large maximum file size with efficient access to large files, but also small files are accessed directly in one disk read.

4.Directory Management

31. cd

# Enter the **test** folder after logging in to the device.

```
<Sysname> cd test
```

# Return to the upper directory (Remember to enter a space after the keyword **cd**).

```
<Sysname> cd ..
```

# Return to the root directory.

```
<Sysname> cd /
```

After you change the current directory using the **cd** command, you can use the **pwd** command to view the path of the current working directory.

32. copy

# Copy file **testcfg.cfg** in the current folder and save it as **testbackup.cfg**.

```
<Sysname> copy testcfg.cfg testbackup.cfg
Copy cfa0:/test.cfg to cfa0:/testbackup.cfg?[Y/N]:y
....
%Copy file cfa0:/test.cfg to cfa0:/testbackup.cfg...Done.
```

# Copy file **1.cfg** in the **test** folder on the first partition of the CF card to the **testbackup** folder on the second partition of the CF card, and save it as **1backup.cfg**.

```
<Sysname> copy cfa0:/test/1.cfg cfa1:/testbackup/1backup.cfg
Copy cfa0:/test/1.cfg to cfa1:/testbackup/1backup.cfg?[Y/N]:y

%Copy file cfa0:/test/1.cfg to cfa1:/testbackup/1backup.cfg...Done.
```

33. dir

# Display information about all files and folders.

```
<Sysname> dir /all
Directory of cfa0:/

  0   drw-   6985954  Apr 26 2007 21:06:29   logfile
  1   -rw-      1842  Apr 27 2007 04:37:17   mainup.app
  2   -rw-      1518  Apr 26 2007 12:05:38   config.cfg
  3   -rw-      2045  May 04 2007 15:50:01   backcfg.cfg
  4   -rwh       428  Apr 27 2007 16:41:21   hostkey
  5   -rwh       572  Apr 27 2007 16:41:31   serverkey
  6   -rw-   2737556  Oct 12 2007 01:31:44   [old.app]

14605 KB total (5096 KB free)
[ ] indicates this file is in the recycle bin.
```

Table 1 **dir** command output description

Field	Description
Directory of	The current working directory
d	Indicates a directory; if this field does not exist, it indicates a file.
r	Indicates that the file or directory is readable.
w	Indicates that the file or directory is writable.
h	Indicates that the file or directory is hidden.
[ ]	Indicates that the file is in the recycle bin.

34. mkdir

# Create a folder named **test** in the current directory.

```
<Sysname> mkdir test
....
%Created dir cfa0:/test
```

# Create folder **test/subtest** in the current directory.

```
<Sysname> mkdir test/subtest
....
%Created dir cfa0:/test/subtest
```

35. more

# Display the contents of file **test.txt**.

```
<Sysname> more test.txt
Welcome to H3C.
```

# Display the contents of file **testcfg.cfg**.

```
<Sysname> more testcfg.cfg

#
 version 5.20, Beta 1201, Standard
#
 sysname Sysname
#
vlan 2
#
return
<Sysname>
```

36. move

# Move file **cfa0:/test/sample.txt** to **cfa0:/**, and save it as **1.txt**.

```
<Sysname> move test/sample.txt 1.txt
Move cfa0:/test/sample.txt to cfa0:/1.txt?[Y/N]:y
...
% Moved file cfa0:/test/sample.txt to cfa0:/1.txt
```

# Move file **b.cfg** to the subfolder **test2**.

```
<Sysname> move b.cfg test2
Move cfa0:/b.cfg to cfa0:/test2/b.cfg?[Y/N]:y
.
%Moved file cfa0:/b.cfg to cfa0:/test2/b.cfg.
```

37. pwd

# Display the current path.

```
<Sysname> pwd
cfa0:
```

38. rename

# Rename file **sample.txt** as **sample.bat**.

```
<Sysname> rename sample.txt sample.bat
Rename cfa0:/sample.txt to cfa0:/sample.bat? [Y/N]:y

% Renamed file cfa0:/sample.txt to cfa0:/sample.bat
```

39. rmdir

# Remove folder **mydir**.

```
<Sysname> rmdir mydir
Rmdir cfa0:/mydir? [Y/N]:y

%Removed directory cfa0:/mydir.
```

5.File System

40. What do you mean a File System?

File System is a method to store and organize files and directories on disk. A file system can have different formats called file system types. These formats determine how the information is stored as files and directories.

41. How do you get its usage (a filesystem)?
 By storing and manipulate files.

42. How do you list files in a directory?
    ls - list directory contents

43. What is the location of "network" file and what does this contains?

location :- /etc/sysconfig/network
This file contains following fields
NETWORKING=yes
NETWORKING_IPV6=no
HOSTNAME=localhost.localdomain

44. What would you use to view contents of the file?
    less filename
    cat filename
    pg filename
    pr filename
    more filename
    most useful is command: tail file_name - you can see the end of the log file.

45. Which deamon is required to start Network services?

network
/etc/init.d/network start

46. Tell me the name of directory structure hierarchy for Linux

/root
/boot
/bin
/sbin
/proc
/mnt
/usr
/var
/lib
/etc
/dev
/opt
/srv
/tmp
/media

47. What does /boot directory contains?

The /boot/ directory contains static files required to boot the system, such as the Linux kernel, boot loader configuration files. These files are essential for the system to boot properly.

48. How you will install software by YUM?

yum install <pkgname>

49. If some one deletes /boot directory from your server, than what will happen?

In that case your server will be in unbootable state. Your Server can't boot without /boot directory because this directory contains all bootable files

50. What is YUM?

YUM stands for Yellow dog Updater, Modified because it is based on YUP, the Yellow dog Updater. Where does the name Yellow dog come from? Yellow Dog is a version of Linux for the Power Architecture hardware and is RPM-based, just like Red Hat Enterprise Linux and Fedora. YUP, and later YUM, were written by the Linux community as a way to maintain an RPM-based system.

51. What does /dev directory contain?

The /dev directory contains all device files that are attached to system or virtual device files that are provided by the kernel.

52. What is the role of udev daemon?

The udev demon used to create and remove all these device nodes or files in /dev/ directory.

6.Network and Communication commands

53. **ping**
The ping command sends an echo request to a host available on the network. Using this command you can check if your
remote host is responding well or not.
The ping command is useful for the following:

- Tracking and isolating hardware and software problems.
- Determining the status of the network and various foreign hosts.
- Testing, measuring, and managing networks

```
$ping google.com
PING google.com (74.125.67.100) 56(84) bytes of data.
64 bytes from 74.125.67.100: icmp_seq=1 ttl=54 time=39.4 ms
64 bytes from 74.125.67.100: icmp_seq=2 ttl=54 time=39.9 ms
64 bytes from 74.125.67.100: icmp_seq=3 ttl=54 time=39.3 ms
64 bytes from 74.125.67.100: icmp_seq=4 ttl=54 time=39.1 ms
64 bytes from 74.125.67.100: icmp_seq=5 ttl=54 time=38.8 ms
--- google.com ping statistics ---
22 packets transmitted, 22 received, 0% packet loss, time 21017ms
rtt min/avg/max/mdev = 38.867/39.334/39.900/0.396 ms
$
```

54. **ftp**
Here ftp stands for **F**ile **T**ransfer **P**rotocol. This utility helps you to upload and download your file from one computer to another computer.
The ftp utility has its own set of UNIX like commands which allow you to perform tasks such as:

- Connect and login to a remote host.
- Navigate directories.

- List directory contents
- Put and get files
- Transfer files as ascii, ebcdic or binary

```
$ftp amrood.com
Connected to amrood.com.
220 amrood.com FTP server (Ver 4.9 Thu Sep 2 20:35:07 CDT 2009)
Name (amrood.com:amrood): amrood
331 Password required for amrood.
Password:
230 User amrood logged in.
ftp> dir
200 PORT command successful.
150 Opening data connection for /bin/ls.
total 1464
drwxr-sr-x   3 amrood   group       1024 Mar 11 20:04 Mail
drwxr-sr-x   2 amrood   group       1536 Mar  3 18:07 Misc
drwxr-sr-x   5 amrood   group        512 Dec  7 10:59 OldStuff
drwxr-sr-x   2 amrood   group       1024 Mar 11 15:24 bin
drwxr-sr-x   5 amrood   group       3072 Mar 13 16:10 mpl
-rw-r--r--   1 amrood   group     209671 Mar 15 10:57 myfile.out
drwxr-sr-x   3 amrood   group        512 Jan  5 13:32 public
drwxr-sr-x   3 amrood   group        512 Feb 10 10:17 pvm3
226 Transfer complete.
```

55. **Telnet**

Many times you would be in need to connect to a remote Unix machine and work on that machine remotely. Telnet is a utility that allows a computer user at one site to make a connection, login and then conduct work on a computer at another site.

Once you are login using telnet, you can perform all the activities on your remotely connect machine. Here is example telnet session:

```
C:>telnet amrood.com
Trying...
Connected to amrood.com.
Escape character is '^]'.

login: amrood
amrood's Password:
*****************************************************
*                                                   *
*                                                   *
*    WELCOME TO AMROOD.COM                          *
*                                                   *
*                                                   *
*****************************************************

Last unsuccessful login: Fri Mar  3 12:01:09 IST 2009
Last login: Wed Mar  8 18:33:27 IST 2009 on pts/10

   {  do your work }
```

```
$ logout
Connection closed.
C:>
```

56. Finger

The finger command displays information about users on a given host. The host can be either local or remote.

Finger may be disabled on other systems for security reasons.

Following are the simple syntax to use finger command:

Check all the logged in users on local machine as follows:

```
$ finger
Login     Name       Tty      Idle  Login Time   Office
amrood               pts/0          Jun 25 08:03 (62.61.164.115)
```

Get information about a specific user available on local machine:

```
$ finger amrood
Login: amrood                          Name: (null)
Directory: /home/amrood                    Shell: /bin/bash
On since Thu Jun 25 08:03 (MST) on pts/0 from 62.61.164.115
No mail.
No Plan.
```

Check all the logged in users on remote machine as follows:

```
$ finger @avtar.com
Login     Name       Tty      Idle  Login Time   Office
amrood               pts/0          Jun 25 08:03 (62.61.164.115)
```

Get information about a specific user available on remote machine:

```
$ finger amrood@avtar.com
Login: amrood                          Name: (null)
Directory: /home/amrood                    Shell: /bin/bash
On since Thu Jun 25 08:03 (MST) on pts/0 from 62.61.164.115
No mail.
No Plan.
```

57. netstat

Displays contents of /proc/net files. It works with the Linux Network Subsystem, it will tell you what the status of ports are ie. open, closed, waiting, masquerade connections. It will also display various other things. It has many different options.

58. hostname

Tells the user the host name of the computer they are logged into. Note: may be called *host.*

59. traceroute

traceroute will show the route of a packet. It attempts to list the series of hosts through which your packets travel on their way to a given destination. Also have a look at *xtraceroute* (one of several graphicalequivalents of this program).

Command syntax:

```
traceroute machine_name_or_ip
```

60. **ifconfig**

This command is used to configure network interfaces, or to display their current configuration. In addition to activating and deactivating interfaces with the "up" and "down" settings, this command is necessary for setting an interface's address information if you don't have the *ifcfg* script.

Use *ifconfig* as either:

```
ifconfig
```

This will simply list all information on all network devices currently up.

```
ifconfig eth0 down
```

This will take eth0 (assuming the device exists) down, it won't be able to receive or send anything until you put the device back "up" again.

Clearly there are a lot more options for this tool, you will need to read the manual/info page to learn more about them.

61. **route**

The *route* command is the tool used to display or modify the routing table. To add a gateway as the default you would type:

```
route add default gw some_computer
```

7. Filter Commands

62. Write a command to print the lines that has the the pattern "july" in all the files in a particular directory?

    ```
    grep july *
    ```
    This will print all the lines in all files that contain the word "july" along with the file name. If any of the files contain words like "JULY" or "July", the above command would not print those lines.

63. How to run the last executed find command?

    ```
    !find
    ```

64. Write a command to print the lines that has the word "july" in all the files in a directory and also suppress the filename in the output.

    ```
    grep -h july *
    ```

65. How to find for a file using name?

```
     find -name "sum.java"
./bkp/sum.java
./sum.java
```

66. Write a command to print the lines that has the word "july" while ignoring the case.

grep -i july *
The option i make the grep command to treat the pattern as case insensitive.

67. When you use a single file as input to the grep command to search for a pattern, it won't print the filename in the output. Now write a grep command to print the filename in the output without using the '-H' option.
grep pattern filename /dev/null
The /dev/null or null device is special file that discards the data written to it. So, the /dev/null is always an empty file.

Another way to print the filename is using the '-H' option. The grep command for this is
grep -H pattern filename

68. Write a Unix command to display the lines in a file that do not contain the word "july"?
grep -v july filename
The '-v' option tells the grep to print the lines that do not contain the specified pattern.

69. Write a command to print the file names in a directory that has the word "july"?
grep -l july *
The '-l' option make the grep command to print only the filename without printing the content of the file. As soon as the grep command finds the pattern in a file, it prints the pattern and stops searching other lines in the file.

70. Write a command to print the line numbers along with the line that has the word "july"?
grep -n july filename
The '-n' option is used to print the line numbers in a file. The line numbers start from 1

71. Write a command to print the lines that starts with the word "start"?
grep '^start' filename
The '^' symbol specifies the grep command to search for the pattern at the start of the line.

8.Archive Commands

72. **Unix tar command line options**

c -- create, for creating tar file
v -- verbose, display name of files including,excluding from tar command
f -- following, used to point name of tar file to be created. it actually tells tar command that name of the file is "next" letter just after options.

x -- extract, for extracting files from tar file.
t -- for viewing content of tar file
z -- zip, tells tar command that create tar file using gzip.
j –- another compressing option tells tar command to use bzip2 for compression
r -- update or add file or directory in already existed .tar file
wildcards -- to specify patters in unix tar command

73. **How to create tar archive or tar file in Unix**

Most of use use either winzip or winrar in windows machine to zipping or creating archives of content so when we move to command line interface like Unix or Linux we struggle without those tools. UNIX tar command is similar to winzip or winrar and you can use UNIX tar command to create both compressed or uncompressed (zipped) archives in UNIX.

In this example of tar command we will create tar file including all the files and directories or selected files and directories in Unix.

here is our directory

```
stock_trader@system:~/test ls -lrt
total 0
-r--r--r--  1 stock_trader Domain Users 0 Jul 15 11:42 equity
drwxrwxrwx+ 1 stock_trader Domain Users 0 Jul 15 14:33 stocks/
-r--r--r--  1 stock_trader Domain Users 0 Jul 15 15:30 currency
```

it has two files and one directory. now we will create a tar file with all these contents.

```
stock_trader@system:~/test tar -cvf trading.tar *
currency
equity
stocks/
stocks/online_stock_exchanges.txt
```

You see unix tar command is creating tar file with name "**trading**" with contents shown above. just to review here "-c" is used to create tar file "v" is used to be verbose and "f" is used to tell tar file name. You can see the tar file here

```
stock_trader@system:~/test ls -lrt
-r--r--r--  1 stock_trader Domain Users   0 Jul 15 11:42 equity
drwxrwxrwx+ 1 stock_trader Domain Users   0 Jul 15 14:33 stocks/
-r--r--r--  1 stock_trader Domain Users   0 Jul 15 15:30 currency
-rw-r--r--  1 stock_trader Domain Users 10K Jul 18 12:29 trading.tar
```

74. **How to view contents of tar file in Unix or Linux**

In earlier example of tar command in Unix or Linux we have created a uncompressed tar file called "trading.tar" now in this example we will see the actual content of that tar file.

```
stock_trader@system:~/test tar -tvf trading.tar
-r--r--r-- stock_trader/Domain Users 0 2011-07-15 15:30 currency
-r--r--r-- stock_trader/Domain Users 0 2011-07-15 11:42 equity
drwxrwxrwx stock_trader/Domain Users 0 2011-07-15 14:33 stocks/
-rwxrwxrwx stock_trader/Domain Users 0 2011-07-15 14:33 stocks/online_stock_exchanges.txt
```

here option "t" is used to display content of tar file in unix while options "v" and "f" are for "verbose" and "following". now you can clearly see that all the files which we wanted to be included in tar file are there.

75. **How to extract contents from a tar file in Unix**

In this example of unix tar command we will see how to extract files or directories from a tar file in unix or Linux. We will use same trading.tar file created in earlier example. In this example we will create a directory "trading" and extract contents of trading.tar on that directory.

```
stock_trader@system:~/test/new ls -lrt
total 12K
-rw-r--r-- 1 stock_trader Domain Users 10K Jul 18 12:37 trading.tar
```

Now the directory is empty just trading.tar file

```
stock_trader@system:~/test/new tar -xvf trading.tar
currency
equity
stocks/
stocks/online_stock_exchanges.txt
```

This unix tar command will extract content of trading.tar in current directory. "x" is used for extracting. "v" is again for verbose and optional parameter in all our example.

```
stock_trader@system:~/test/new ls -lrt
-r--r--r--  1 stock_trader Domain Users   0 Jul 15 11:42 equity
drwxr-xr-x+ 1 stock_trader Domain Users   0 Jul 15 14:33 stocks/
-r--r--r--  1 stock_trader Domain Users   0 Jul 15 15:30 currency
-rw-r--r--  1 stock_trader Domain Users 10K Jul 18 12:37 trading.tar
```

Now you can see that all the files and directories which were included in tar file (stocks, equity and currency) has been extracted successfully.

76. cpio

cpio stands for **Copy in and out**. Cpio is a general purpose file archiver for Linux. It is actively used by **RedHat Package Manager** (RPM) and in the **initramfs** of Linux Kernel as well as an important archiving tool in **Apple Computer's Installer** (pax).

cpio options

- **-0** : Read a list of filenames terminated by a null character instead of a newline.
- **-a** : Reset Access time.
- **-A** : Append.
- **-b** : swap.
- **-d** : Make Directories.

Create an 'cpio' archive file.

```
# cd tecmint
# ls
file1.o file2.o file3.o
# ls | cpio  -ov > /path/to/output_folder/obj.cpio
```

To extract a cpio archive file.

```
# cpio -idv < /path/to folder/obj.cpio
```

77. gzip

gzip is standard and widely used file compression and decompression utility. Gzip allows file concatenation. Compressing the file with gzip, outputs the tarball which is in the format of '***.tar.gz**' or '***.tgz**'.

gzip options

- **–stdout** : Produce output on standard output.
- **–to-stdout** : Produce output on standard output.
- **–decompress** : Decompress File.
- **–uncompress** : Decompress File.
- **-d** : Decompress File.
- **-f** : Force Compression/Decompression.

Create an 'gzip' archive file.

```
# tar -cvzf name_of_archive.tar.gz /path/to/folder
```

To extract a 'gzip' archive file.

```
# gunzip file_name.tar.gz
```

The above command must be passed followed with below command.

```
# tar -xvf file_name.tar
```

Note: The architecture and functionality of 'gzip' makes it difficult to recover corrupted 'gzipped tar archive' file. It is advised to make several backups of gzipped Important files, at different Locations.

That's all for now. We will be discussing other compressing and decompressing applications, available for Linux, in our next article. Till then stay tuned and connected to **Tecmint**. Don't forget to provide us with your valuable feedback in the comment section below.

78. **bzip2**

An alternate compression utility, usually more efficient than **gzip**, especially on large files. The corresponding decompression command is **bunzip2**.

79. **compress**, **uncompress**

This is an older, proprietary compression utility found in commercial UNIX distributions. The more efficient **gzip** has largely replaced it. Linux distributions generally include a **compress** workalike for compatibility, although **gunzip** can unarchive files treated with **compress**.

80. **zip**, **unzip**

Cross-platform file archiving and compression utility compatible with DOS *PKZIP*. "Zipped" archives seem to be a more acceptable medium of exchange on the Internet than "tarballs".

81. **shar**

Shell archiving utility. The files in a shell archive are concatenated without compression, and the resultant archive is essentially a shell script, complete with #!/bin/sh header, and containing all the necessary unarchiving commands. Shar archives still show up in Internet newsgroups, but otherwise **shar** has been pretty well replaced by **tar**/**gzip**. The **unshar** command unpacks **shar** archives.

82. **ar**

Creation and manipulation utility for archives, mainly used for binary object file libraries.

9. Text Manipulation And Searching Commands

83. **What is sed?**
    - sed is **s**tream **ed**itor, a Unix tool for working with streams of text data.

84. **How do you substitute strings with sed?**

- Use 's/old/new' command, so sed 's/hello/goodbye/' would substitute the occurrence of the word hello to goodbye.

85. **How do you search for a string inside a given file?**

- grep string filename

86. **How do you search for a string inside a directory?**

- grep string *

87. **How do you search for a string in a directory with the subdirectories recursed?**

- grep -r string *

88. **How do you inject text with sed?**

- & in the substitution string defines the pattern found in the search string. As an example, here's us trying to find a word 'hello' and replacing it with 'hello and how are you':

```
echo 'hello there' | sed 's/^hello/& and how are you/'
```

89. **Can I find several patterns and refer to them in the replacement string**?

- Yes, use (pattern) and then refer to your patterns as \1, \2, \3 and so on.

90. **If the string is 'old old old' and I run 's/old/new', I get 'new old old' as the result. I need 'new new new'**.

- You forgot the global modifier, which would replace every occurrence of the pattern with the substitution. 's/old/new/**g**' will work.

91. **But I want 'old old new' from the previous example**.

- Just use the numeric modifier saying you want the third occurrence to be replaced. 's/old/new/**3**' will work.

92. **I wrote a rather complex sed script. How do I save and run it**?

- Assuming that your file is named myscript1.sed, you can invoke sed -f myscript1.sed.

93. **How do I delete trailing whitespaces from each line**?

- sed ’s/[\t]*$//’ Here we’re replacing any occurrence of a space or a tab with nothing. Check sed one-liners for more examples.

94. **How do you print just a few first lines of the file**?
- sed 1q will give you just the first line, sed 10q the first 10 lines.

95. **How do you replace a pattern only if it’s found, so that it’s executed faster**?

- Nest the replacement statement: sed ‘/old/ s/old/new/g’ file.txt

10. Process and Job Controller

96. Write a command to kill the last background job?
    Kill $!

97. What is ‘ps’ command for?
    The ps command prints the process status for some or all of the running processes. The information given are the process identification number (PID),the amount of time that the process has taken to execute so far etc.

98. How would you kill a process?
    The kill command takes the PID as one argument; this identifies which process to terminate. The PID of a process can be got using ‘ps’ command.

99. What is an advantage of executing a process in background?
    The most common reason to put a process in the background is to allow you to do something else interactively without waiting for the process to complete. At the end of the command you add the special background symbol, &. This symbol tells your shell to execute the given command in the background.
    Example: cp *.* ../backup& (cp is for copy)

100. Brief about the initial process sequence while the system boots up.
    While booting, special process called the 'swapper' or 'scheduler' is created with Process-ID 0. The swapper manages memory allocation for processes and influences CPU allocation. The swapper inturn creates 3 children:

    the process dispatcher,
    vhand and
    dbflush

    with IDs 1,2 and 3 respectively.

    This is done by executing the file "/etc/init". Process dispatcher gives birth to the shell. Unix keeps track of all the processes in an internal data structure called the Process Table (listing command is ps -el).

101. What are various IDs associated with a process?
    Unix identifies each process with a unique integer called ProcessID. The process that

executes the request for creation of a process is called the 'parent process' whose PID is 'Parent Process ID'. Every process is associated with a particular user called the 'owner' who has privileges over the process. The identification for the user is 'UserID'. Owner is the user who executes the process. Process also has 'Effective User ID' which determines the access privileges for accessing resources like files.

- getpid() -process id
- getppid() -parent process id
- getuid() -user id
- geteuid() -effective user id

102.Explain fork() system call.
The 'fork()' used to create a new process from an existing process. The new process is called the child process, and the existing process is called the parent. We can tell which is which by checking the return value from 'fork()'. The parent gets the child's pid returned to him, but the child gets 0 returned to him.

103.Predict the output of the following program code.
```
main()
{
  fork();
  printf("Hello World!");
}
```
Answer: Hello World!Hello World!
Explanation: The fork creates a child that is a duplicate of the parent process. The child begins from the fork(). All the statements after the call to fork() will be executed twice.(once by the parent process and other by child). The statement before fork() is executed only by the parent process.

104.Predict the output of the following program code
```
main()
{
  fork(); fork(); fork();
  printf("Hello World!");
}
```
Answer: "Hello World" will be printed 8 times.
Explanation: 2^n times where n is the number of calls to fork();

105.List the system calls used for process management:

System calls	Description
fork()	To create a new process
exec()	To execute a new program in a process
wait()	To wait until a created process completes its execution
exit()	To exit from a process execution
getpid()	To get a process identifier of the current process
getppid()	To get parent process identifier
nice()	To bias the existing priority of a process
brk()	To increase/decrease the data segment size of a process

106.What is a zombie?
When a program forks and the child finishes before the parent, the kernel still keeps some of its information about the child in case the parent might need it - for example, the parent may need to check the child's exit status. To be able to get this information, the

parent calls 'wait()'; In the interval between the child terminating and the parent calling 'wait()', the child is said to be a 'zombie' (If you do 'ps', the child will have a 'Z' in its status field to indicate this.)

107. What are the process states in Unix?
As a process executes it changes state according to its circumstances. Unix processes have the following states:
Running : The process is either running or it is ready to run .
Waiting : The process is waiting for an event or for a resource.
Stopped : The process has been stopped, usually by receiving a signal.
Zombie : The process is dead but have not been removed from the process table.

108. How can a parent and child process communicate?
A parent and child can communicate through any of the normal inter-process communication schemes (pipes, sockets, message queues, shared memory), but also have some special ways to communicate that take advantage of their relationship as a parent and child. One of the most obvious is that the parent can get the exit status of the child.

11. File Permissions

109. **Linux Read mode permissions**

- Read access on a file allows you to view file
- Read access on a directory allows you to view directory contents with ls command

110. **Write mode permissions**

- Write access on a file allows you to write to file
- Write access on a directory allows you to remove or add new files

111. **Execute mode permissions**

- Execute access on a file allows to run program or script
- Execute access on a directory allows you access file in the directory

112. **Octal numbers and permissions**

You can use octal number to represent mode/permission:

- r: 4
- w: 2
- x: 1

For example, for file owner you can use octal mode as follows. Read, write and execute (full) permission on a file in octal is

0+r+w+x = 0+4+2+1 = 7

Only Read and write permission on a file in octal is

0+r+w+x = 0+4+2+0 = 6

Only read and execute permission on a file in octal is

0+r+w+x = 0+4+0+1 = 5

Use above method to calculate permission for group and others. Let us say you wish to give full permission to owner, read & execute permission to group, and read only permission to others, then you need to calculate permission as follows:
User = r+w+x = 0+4+2+1 = 7
Group= r+w+x = 0+4+2+0 = 6
Others = r+w+x = 0+0+0+1 = 1
Effective permission is 761.

113. **chmod command**

To setup file permission you need to use chmod command:
chmod {mode} {file-name}

To setup file permission 761 you need to use chmod command as follows:

```
# chmod 0761 file
```

To setup a file readable by anyone and writable by the owner only:

```
# chmod 644 file
```

To setup a file readable/executable by everyone and writable by the owner only:

```
# chmod 755 file
```

You can change permissions for all files and directories within a directory by using the -R option on the chmod command. For example, to setup others read and execute access to all files and directories (and files and directories within directories), you need to type command as follows (i.e. change the modes of the file hierarchies rooted in the files instead of just the files themselves):

```
# chmod -R 755 directory-name/
```

114. Give execute privilege to user. Leave other privileges untouched

execute = 1. If you want to just add execute privilege to users and leave all other privileges as it is, do the following.

```
$ chmod u+x file.txt
```

115. Give read, write and execute to everybody (user, group, and others)

read, write and execute = 4 + 2 + 1 = 7.

```
$ chmod 777 file.txt
(or)
$ chmod ugo+rwx file.txt
```

116. Give read, write and execute privileges to group (including all the files in the sub-directories)

Use -R, as shown below to provide the recursive privileges for the directory and sub-directories (including the files in it).

```
$ chmod -R g+rwx /u01
```

117. Clear the user permission bit of **example.jpg** and set it to write-only.

chmod u=w example.jpg

118. Set the User ID bit of **comphope.txt**, so that anyone who attempts to access that file does so as if they are the owner of the file.

chmod u+s comphope.txt

12. Vi Editor

119. Explain What does the /text command do?
/text: it will search for the string. after pressing enter
it takes u to that text location.

120. Which command is used to replace many characters in Vi Editor?
change command can be used to change a word/line.

cw change word forward
cb change word backward
c$ change from cursor to end of line
cL change from current line to and of screen
cG change from current line to and of file

or if you want to replace all occurence of some specific
character

:%s/oldText/newText/g

121. What is the difference between lettered buffer and temporary buffer in Vi Editor?
Temporary Buffer

Deleted or copied text goes into a temporary unnamed
buffer. The contents of the temporary buffer may be
retrieved by using the p or P commands.

Lettered Buffers

There are 26 lettered buffers (a-z). Contents of a lettered
buffer are saved until you copy or delete more characters
into it, or until you quit your current vi session.

eg.
From Command Mode

"ayy Copy (yank) a line into buffer letter "a"
"ap Put contents of lettered buffer a below the
current line

122. what are the different modes in vi editor?
There are three basic modes of vi:

Command mode
This is the default when you enter vi. In command mode,
most letters, or short sequences of letters, that you type
will be interpreted as commands, without explicitly
pressing Enter . If you press Esc when you're in command
mode, your terminal will beep at you. This is a very good

way to tell when you're in command mode.

Insert mode
In insert mode, whatever you type is inserted in the file at the cursor position. Type a (lowercase letter a, for append) to enter insert mode from command mode; press Esc to end insert mode, and return to command mode.

Line mode
Use line mode to enter line oriented commands. To enter line mode from command mode, type a colon (:). Your cursor moves to the bottom of the screen, by a colon prompt. Type a line mode command, then press Enter. Any sensible command from the Unix line editor ex will work, and a few are good to know about. These commands are indicated in this handout by a colon in front of the command. Each time you use a line mode command, you must type a colon to enter line mode, then type the command by the colon prompt at the bottom of the screen, then press Enter when you finish typing the command. (The search commands starting with / and ? work similarly.

123. what is the command used to append text after current line in Vi Editor?
a

124. How to enter from command mode to insertion mode using Vi Editor?
There are several commands that put the VI editor into insert mode. The most commonly used commands to get into insert mode are a and i.

125. What is the command used to replace many characters in Vi Editor?
for replace many character in vi editor press esc key and then press R for replace many character.

126. What is the difference between ZZ and :wq commands in Vi Editor?
ZZ is the command mode comand in uix to save and quit file.
:wq is the execute command mode command to save and quit file.

127. Explain What is the command used to set margin in vi editor?
Press ESC followed by colon setnu

eg => :setnu

128. Explain What is the format of vi command?
vi filename

129. How to append a file to current file using Vi Editor?
Alternate approach:

If you are working in file2 and want to append file1, than place the cursor where you want to append the new file and use the following command

:r file1

130. What does the c$ command do from command mode using Vi Editor?
c$ will begin from the character under the curser till the
end of line. so when you use this command it will show you $
sign at the end of the line and you can change till that point.

13. Cron Jobs

131. Syntax of crontab (field description)

1 2 3 4 5 /path/to/command arg1 arg2

Where,

1: Minute (0-59)

2: Hours (0-23)

3: Day (0-31)

4: Month (0-12 [12 == December])

5: Day of the week(0-7 [7 or 0 == sunday])

/path/to/command - Script or command name to schedule

132. Easy way to remember the cron job syntax

```
* * * * * command to be executed
- - - - -
| | | | |
| | | | ----- Day of week (0 - 7) (Sunday=0 or 7)
| | | ------- Month (1 - 12)
| | --------- Day of month (1 - 31)
| ----------- Hour (0 - 23)
------------- Minute (0 - 59)
```

133. To run /path/to/command five minutes after midnight, every day, enter:

```
5 0 * * * /path/to/command
```

134. Run /scripts/phpscript.php at 10 pm on weekdays, enter:

```
0 22 * * 1-5 /scripts/phpscript.php
```

135. Run /path/to/script.sh at 2:15pm on the first of every month, enter:

```
15 14 1 * * /path/to/script.sh
```

136. Run /root/scripts/perl/perlscript.pl at 23 minutes after midnight, 2am, 4am ..., everyday, enter:

```
23 0-23/2 * * * /root/scripts/perl/perlscript.pl
```

137. Run /path/to/unixcommand at 5 after 4 every Sunday, enter:

```
5 4 * * sun /path/to/unixcommand
```

138. How to list all tha cron jobs

```
# crontab -l
# crontab -u username -l
```

139. To remove or erase all crontab jobs use the following command:
crontab -r

140. Delete job for specific user
crontab -r -u username

141. Special strings to save time

Special string	Meaning
@reboot	Run once, at startup.
@yearly	Run once a year, "0 0 1 1 *".
@annually	(same as @yearly)
@monthly	Run once a month, "0 0 1 * *".
@weekly	Run once a week, "0 0 * * 0".
@daily	Run once a day, "0 0 * * *".
@midnight	(same as @daily)
@hourly	Run once an hour, "0 * * * *".

142. **How do I backup installed cron jobs entries?**
Simply type the following command to backup your cronjobs to a nas server mounted at /nas01/backup/cron/users.root.bakup directory:

```
# crontab -l > /nas01/backup/cron/users.root.bakup
# crontab -u userName -l > /nas01/backup/cron/users.userName.bakup
```

14. Environment Variables And Path

143. Following are most command examples of environment variables used under UNIX operating systems:

PATH - Display lists directories the shell searches, for the commands.

- HOME - User's home directory to store files.
- TERM - Set terminal emulator being used by UNIX.
- PS1 - Display shell prompt in the Bourne shell and variants.
- MAIL - Path to user's mailbox.
- TEMP - Path to where processes can store temporary files.
- JAVA_HOME - Sun (now Oracle) JDK path.
- ORACLE_HOME - Oracle database installation path.
- TZ - Timezone settings
- PWD - Path to the current directory.
- HISTFILE - The name of the file in which command history is saved

- HISTFILESIZE -The maximum number of lines contained in the history file
- HOSTNAME -The system's host name
- LD_LIBRARY_PATH -It is a colon-separated set of directories where libraries should be searched for.
- USER -Current logged in user's name.
- DISPLAY -Network name of the X11 display to connect to, if available.
- SHELL -The current shell.
- TERMCAP - Database entry of the terminal escape codes to perform various terminal functions.
- OSTYPE - Type of operating system.
- MACHTYPE - The CPU architecture that the system is running on.
- EDITOR - The user's preferred text editor.
- PAGER - The user's preferred text pager.
- MANPATH - Colon separated list of directories to search for manual pages.

144.Open the terminal and type the following commands to display all environment variables and their values under UNIX-like operating systems:

```
$ set
or
$ printenv
or
$ env
```

145.To display search path, enter:

```
echo $PATH
```

146.To display prompt settings, enter:

```
echo $PS1
```

147.To set JAVA_PATH, enter:

```
JAVA_PATH=/opt/jdk/bin
export JAVA_PATH
```

148.ostype
An example of an environment variable is the OSTYPE variable. The value of this is the current operating system you are using. Type

```
% echo $OSTYPE
```

149. To show all values of these variables, type

```
% printenv | less
```

150.history
An example of a shell variable is the history variable. The value of this is how many shell commands to save, allow the user to scroll back through all the commands they have previously entered. Type

% echo $history

151. To change the number of shell commands saved in the history list, you need to set the shell variable history. It is set to 100 by default, but you can increase this if you wish.

% set history = 200

152. **Displaying the Value of an Environment Variable**

To display the values of individual environment variables, use the **echo** command and parameter substitution. An example is: **echo $SHELL** which returns the current value of the SHELL environment variable. Use the **env** (or **printenv**) command to display all environment variables and their current values.

15. Shell Scripting

153. How do you read keyboard input in shell scripts?

- read {variable-name}

154. How do you define a function in a shell script?

- function-name() { #some code here return }

155. How does a case statement look in shell scripts?

- case {variable} in {possible-value-1}) {statement};; {possible-value-2}) {statement};; esac

156. How do you write a while loop in shell?

- while {condition} do {statement} done

157. How do you write a for loop in shell?

- for {variable name} in {list} do {statement} done

158. What's a way to do multilevel if-else's in shell scripting?

- if {condition} then {statement} elif {condition} {statement} fi

159. How do you find out the number of arguments passed to the shell script?

- $#

160. How do you do Boolean logic operators in shell scripting?

- ! tests for logical not, -a tests for logical and, and -o tests for logical or.

161. How do you test for file properties in shell scripts?

- -s filename tells you if the file is not empty, -f filename tells you whether the argument is a file, and not a directory, -d filename tests if the argument is a directory, and not a file, -w filename tests for writeability, -r filename tests for readability, -x filename tests for executability

162. How do you do number comparison in shell scripts?

- -eq, -ne, -lt, -le, -gt, -ge

163. What's the conditional statement in shell scripting?

- if {condition} then … fi

164. How do you refer to the arguments passed to a shell script?

- $1, $2 and so on. $0 is your script name.