

FREE AND OPEN SOURCE SOFTWARE LAB MANUAL



SEMESTER IV

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
COLLEGE OF ENGINEERING, TRIVANDRUM

Vision and Mission of the Institution

Vision

National level excellence and international visibility in every facet of engineering education.

Mission

To facilitate quality engineering education to equip and enrich young men and women to meet global challenges in development, innovation and application of technology in the service of humanity.

Vision and Mission of the Department

Vision

To be a centre of excellence in education and research in the frontier areas of Computer Science and Engineering.

Mission

To facilitate quality education and research in Computer Science and Engineering, for transforming students into competent professionals catering to needs of academia, industry and society.

Program Outcomes

PO1 - Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2 - Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3 -Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4 - Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5 - Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO6 -The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7 - Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8 - Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9 - Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10 - Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11 - Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as

a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12 -Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Course Outcomes

Course Code: CS232

Course Name: Free and Open Source Lab

At the end of this course, students will be able to,

CO1 Identify and apply various Linux commands (B3).

CO2 Develop shell scripts for different applications (B3).

CO3 Experiment with distributed version control system tools (B3).

CO4 Experiment with text processing using Perl and Linux commands (B3).

CO5 Develop simple applications and deploy using LAMP (B3).

CO6 Experiment with kernel configuration, network configuration, packet management and installations (B3).

CO7 Develop simple GUI application using GTK / QT /Gambas (B3).

CO8 Experiment with isolation of application using virtualizations (B3).

CO-PO Mapping

COs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CO1	2	1	1	1	1	1				1					
CO2	3	2	3	2	2	2		2	1	1		1	2		3
CO3	3		3		3	3	2		1	1		1		2	1
CO4	3	2	2	1	3	3			1	1		1	1	2	1
CO5	3	1	3	2	3	3		1	2	1		2	2	3	3
CO6	3	1	2	2	3	3			1	1		2	1	1	3
CO7	3	2	3	2	3	2		2	2	1		2	1	2	2
CO8	3	2	2	2	3	3	2	1	1	1		2	1	2	3
CO	3	2	3	2	3	3	2	2	2	1		2	2	3	3

Course Code	Course Name	L-T-P-Credits	Year of Introduction
CS232	Free and Open Source Software Lab	0-0-3-1	2016
Pre-requisite: CS204 Operating systems			
Course Objectives <ul style="list-style-type: none"> • To expose students to FOSS environment • To introduce them to use open source packages in open source platform. 			
List of Exercises/ Experiments: <ol style="list-style-type: none"> 1. Getting started with Linux basic commands for directory operations, displaying directory structure in tree format etc. 2. Linux commands for operations such as redirection, pipes, filters, job control, changing ownership/permissions of files/links/directory. 3. Advanced linux commands curl, wget, ftp, ssh and grep 4. Shell Programming : Write shell script to show various system configuration like <ul style="list-style-type: none"> • Currently logged user and his login name • Your current shell • Your home directory • Your operating system type • Your current path setting • Your current working directory • Number of users currently logged in 5. Write shell script to show various system configurations like <ul style="list-style-type: none"> • your OS and version, release number, kernel version • all available shells • computer CPU information like processor type, speed etc • memory information • hard disk information like size of hard-disk, cache memory, model etc • File system (Mounted) 6. Write a shell script to implement a menu driven calculator with following functions <ol style="list-style-type: none"> (a) Addition (b) Subtraction (c) Multiplication (d) Division (e) Modulus 7. Write a script called addnames that is to be called as follows <code>./addnames ulist username</code> Here <i>ulist</i> is the name of the file that contains list of user names and <i>username</i> is a particular student's username. The script should <ul style="list-style-type: none"> • check that the correct number of arguments was received and print a message, in case the number of arguments is incorrect • check whether the ulist file exists and print an error message if it does not • check whether the username already exists in the file. If the username exists, print a message stating that the name already exists. Otherwise, add the username to the end of the list. 			

8. Version Control System setup and usage using GIT. Try the following features.
 - Creating a repository
 - Checking out a repository
 - Adding content to the repository
 - Committing the data to a repository
 - Updating the local copy
 - Comparing different revisions
 - Revert
 - Conflicts and a conflict Resolution
9. Shell script which starts on system boot up and kills every process which uses more than a specified amount of memory or CPU.
10. Introduction to packet management system : Given a set of RPM or DEB, build and maintain, and serve packages over http or ftp. Configure client systems to access the package repository.
11. Perform simple text processing using Perl, Awk.
12. Running PHP : simple applications like login forms after setting up a LAMP stack
13. Virtualisation environment (e.g., xen, qemu, virtualbox or lguest) to test applications, new kernels and isolate applications. It could also be used to expose students to other alternate OS such as FreeBSD
14. Compiling from source : learn about the various build systems used like the auto* family, cmake, ant etc. instead of just running the commands. This could involve the full process like fetching from a cvs and also include autoconf, automake etc.,
15. Kernel configuration, compilation and installation : Download / access the latest kernel source code from kernel.org, compile the kernel and install it in the local system. Try to view the source code of the kernel
16. GUI Programming: Create scientific calculator - using any one of Gambas, GTK, QT
17. Installing various software packages. Either the package is yet to be installed or an older version is present. The student can practice installing the latest version. (Internet access is needed).
 - Install samba and share files to windows
 - Install Common Unix Printing System(CUPS)
18. Set up the complete network interface by configuring services such as gateway, DNS, IP tables etc. using ifconfig

Expected Outcome

The students will be able to:

1. Identify and apply various Linux commands
2. Develop shell scripts and GUI for specific needs
3. Use tools like GIT
4. Perform basic level application deployment, kernel configuration and installation, packet management and installation etc.

Evaluation criteria

For Laboratory/ Practical/ Workshop courses:

- **Practical records/ outputs** 60 marks (Internally by the College)
- **Regular class Viva** 10 marks (Internally by the College)
- **Final written test/ quiz** 30 marks (Internally by the College)

All the above assessments are mandatory to earn credits. If not, the student has to complete the course/ assessments during his free time in consultation with the faculty members. On completion of these, grades will be assigned. In case the Practical/ Laboratory/ Workshop courses are not completed in the semester, grade I (incomplete) will be awarded against the course and the final grade will be given only after the completion of the course/ assessments.

Contents

1	Getting started with Linux basic commands for directory operations	11
1.1	Aim	11
1.2	Overview	11
1.3	Sample input/output	13
1.4	Result/Observation	13
1.5	Viva	13
2	Linux commands for operations such as redirection, pipes, filters, job control, changing ownership/permissions of files/links/directory.	15
2.1	Aim	15
2.2	Overview	15
2.2.1	Redirection of standard Input/Output	15
2.2.2	Pipes	15
2.2.3	Filters	15
2.2.4	Job Control	16
2.2.5	Permissions	17
2.3	Sample input/output	17
2.3.1	Link	17
2.4	Result/Observation	19
2.5	Viva	19
3	Advanced Linux Commands	20
3.1	Aim	20
3.2	Overview	20
3.2.1	curl	20
3.2.2	ssh	20
3.2.3	wget	21
3.2.4	ftp	22
3.2.5	grep	22
3.3	Sample input/output	23
3.4	Result/Observation	24
3.5	Viva	24
4	Shell programming	26
4.1	Aim	26
4.2	Overview	26
4.2.1	Shellscript	26
4.3	Sample input/output	26
4.4	Result/Observations	26
4.5	Viva	27
5	Shell script to show various system configurations	28

5.1	Aim	28
5.2	Overview	28
5.3	Sample input/output	28
5.3.1	Shell Script	28
5.4	Result/Observation	28
5.5	Viva	28
6	Menu driven calculator	30
6.1	Aim	30
6.2	Overview	30
6.2.1	Shell script	30
6.3	Sample input/output	31
6.4	Result/Observation	31
6.5	Viva	31
7	Script that accepts two arguments from the commad line and operates on them	33
7.1	Aim	33
7.2	Overview	33
7.3	Sample input/output	34
7.4	Result/Observation	34
7.5	Viva	34
8	Version Control System setup and usage using Git	35
8.1	Aim	35
8.2	Overview	35
8.3	Result/Observation	38
8.4	Viva	38
9	Shell script which starts on system boot up and kills every process which uses more than a specified amount of memory or CPU	40
9.1	Aim	40
9.2	Overview	40
9.3	Sample input/output	41
9.4	Result/Observation	41
9.5	Viva	41
10	Introduction to packet management system	42
10.1	Aim	42
10.2	Overview	42
10.3	Result/Observation	44
10.4	Viva	44
11	Simple Text Processing using Perl and Awk	45

11.1	Aim	45
11.2	Overview	45
11.2.1	Perl	45
11.2.2	Awk	45
11.3	Result/Observation	46
11.4	Viva	46
12	Running PHP	47
12.1	Aim	47
12.2	Overview	47
12.3	Sample input/output	48
12.3.1	Php - Example	48
12.4	Result/Observation	49
12.5	Viva	49
13	Virtualization Environment	51
13.1	Aim	51
13.2	Overview	51
13.2.1	Algorithm	51
13.3	Result/Observation	57
13.4	Viva	57
14	Compiling From The Source	59
14.1	Aim	59
14.2	Overview	59
14.2.1	Tools included in the GNU build system	59
14.2.2	GNU Autoconf	59
14.2.3	GNU Automake	60
14.2.4	GNU Libtool	60
14.2.5	Gnulib	60
14.2.6	Make	60
14.2.7	Cmake	61
14.2.8	Apache Ant	61
14.3	Result/Observation	68
14.4	Viva	68
15	Kernel Configuration, Compilation and Installation	69
15.1	Aim	69
15.2	Overview	69
15.2.1	Algorithm:	69
15.3	Result/Observation	74
15.4	Viva	74
16	GUI Programming	75

16.1	Aim	75
16.2	Overview	75
16.2.1	Python code	75
16.3	Sample input/output	77
16.4	Result/Observation	77
16.5	Viva	77
17	Installing Various Software Packages	79
17.1	Aim	79
17.2	Overview	79
17.2.1	Installation of Samba:	79
17.2.2	Testing Communication of Linux with windows:	79
17.2.3	Installation of CUPS	80
17.3	Result/Observation	81
17.4	Viva	81

1 Getting started with Linux basic commands for directory operations

1.1 Aim

Getting started with Linux basic commands for directory operations, displaying directory structure in tree format etc.

1.2 Overview

A directory in Linux is similar to a folder in windows OS. Files are organized in to directories and sub- directories. In Linux, path begins at the root directory which is the top-level of the file system and is represented as a forward slash (/). Forward slash is used to separate directory and file names.

Table 1: Basic Commands

Command	Operation
touch filename	create a new file
mkdir dirname	create a new directory
pwd	prints present working directory
cd dirname	change directory
cat filename	view contents of a file
more filename	view contents of a file one screenful at a time
less filename	similar but faster than more
ls	list files in a directory
ls -l	provide long listing of all the files
ls -l -h	provides sizes in human readable form
ls -F	mark all executables with * and directories with /
ls -a	show all files in the present directory with special dot files
cp file1 file2	copying files
cp -r dirname1 dirname2	copy directories
rm filename	remove a file
rmdir -r dirname	remove a non empty directory

clear	clear the contents of the terminal
locate	search for a specified filename
man commandname	view help of the specified command name
chmod [options] mode filename	change file permissions
chown [options] filename	Change file ownership
kill [options] pid	Kill a process
who [options]	Display who is logged in
top	Display the resources being used in your system
ln [options] source [desitination]	Create a shortcut

Table 2: Directory structure

File name	Content
/bin	Essential user command binaries
/boot	Static files and boot loader
/dev	Device files
/etc	Host specific system configuration
/home	User home directories
/lib	Essential shared libraries and kernel modules
/mnt	Mount point for devices
/opt	Add on application software packages
/sbin	System binaries
/tmp	Temporary files
/usr	User utilities and applications
/var	variable files
/root	home directory for the root user

1.3 Sample input/output

```
[santhisenan@arch fosslabrecord]$ tree -d -L 1 /
/
├── bin -> usr/bin
├── boot
├── dev
├── etc
├── home
├── lib -> usr/lib
├── lib64 -> usr/lib
├── lost+found
├── mnt
├── opt
├── proc
├── root
├── run
├── sbin -> usr/bin
├── srv
├── sys
├── tmp
├── usr
└── var

19 directories
```

Figure 1: Output

1.4 Result/Observation

Several basic linux commands and the directory structure of linux were studied. The directory structure was printed in tree format using tree command on ArchLinux 4.10.11.

1.5 Viva

1. How to create a directory?
(A) rmdir (B) cd dirname (C) mkdir (D) cpdir
2. How to remove a file?
(A) rm filename (B) cat filename (C) less filename (D) more filename
3. what does ls -a command does?
(A) provide long listing of all the files
(B) show all files in the present directory with special dot files
(C) provides sizes in human readable form
(D) list files in a directory

4. What is the command to print present working directory?
(A) pwd (B) cat (C) rm (D) cd
5. How to copy the contents of one file in to another?
(A) cp -r file1 file2 (B) cat file (C) cp file1 file2 (D) rm file

2 Linux commands for operations such as redirection, pipes, filters, job control, changing ownership/permissions of files/links/directory.

2.1 Aim

Linux commands for operations such as redirection, pipes, filters, job control, changing ownership/permissions of files/links/directory.

2.2 Overview

2.2.1 Redirection of standard Input/Output

- By default most command line programs send their output to the standard output which by default displays it on the

commandName > fileName -Overwrites the file with the output of the command

commandName >> fileName - appends the file with the output of the command.

- Most of the command line programs accept its input from the standard input and by default gets its contents from the keyboard. Similar to standard output it can also be redirected.

sort < filename - sort command processes the contents of the file with the name filename.

sort < file1 > file2 processes the contents of file 1 and redirects its output to file 2

2.2.2 Pipes

Pipes are used to redirect the standard output of one command to the standard input of another command.

command1 | command2 the standard output of command 1 is redirected to the standard input of command 2.

2.2.3 Filters

Filters take the standard input and perform an operation upon it and sends the results to the standard output. This can be used to process information in powerful ways.

- *sort* - sorts the standard input and sends the output to standard output.
'sort filename' rearranges each line of file in alphabetical order and outputs it to the standard output.
- *uniq* - Given a sorted stream of data from standard input it removes the duplicate lines of data and returns the result to the standard output.
- *grep* - examines each line of data it receives from standard input and outputs all lines that contains a specific pattern of characters.
'grep "string" new.txt' outputs lines of text in new.txt which contain the word string.
- *fmt* - reads text from standard input and outputs formatted text to standard output.
'fmt filename' formats contents of filename and outputs it in standard output.
- *pr* - Takes data from the standard input and splits the data into pages with page breaks, footers and headers in preparation for printing.
'pr filename' displays the contents of the file one page after the other and returns the output to the standard output.
- *head* - Outputs the first few lines of the file and returns it to the standard output.
- *tail* - Outputs the last few lines of the file and returns it to the standard output.
- *tr* - Translates Characters. Can be used to perform tasks such as uppercase to lowercase conversions or changing the line termination characters from one type to another.
'tr [:lower:] [:upper:]' takes input from the keyboard and outputs each character of the input to uppercase characters and outputs it to the standard output.

2.2.4 Job Control

There are several commands used to control processes in Linux.

- *ps* - The ps commands lists the processes running in the system.
'ps lists' all processes running in the system
'ps aux' lists all processes running in the system.
- *kill* - sends a signal to the specified processes usually to stop the execution of the processes.
'kill -l' lists the signal names that can be sent to processes in Linux.
'kill pid' to kill the processes specified by the process id (pid) which can be obtained by the ps command.
'kill -s SIGKILL pid' is used to send SIGKILL signal to process with process id 'pid'. This command is used to forcefully kill a process without memory cleanup.

A signal is an asynchronous notification sent to a process or to a specific thread within the same process in order to notify it of an event that occurred.

- *jobs* - An alternate way of listing the processes. *jobs* is a shell builtin command which gives you information internal to the shell such as the job numbers. ‘jobs’ lists the jobs that the current shell is managing.
- *bg* - used to put a process in background.
- *fg* - used to put a process in foreground.

2.2.5 Permissions

Table 3: Permissions

chmod	modify file access rights
su	temporarily become super user
chown	change file ownership
chgrp	change file group ownership

2.3 Sample input/output

2.3.1 Link

A link provides a connection between files. This provides the ability to have a single file or directory referred to through different names.

The nearest comparison with the Windows world is of a shortcut, but that is an unfair comparison as a link in Linux is far more powerful. A Windows shortcut is just a way of launching a file from a different place, whereas a link can make a file appear in multiple locations which is invisible to the applications.

There are two types of links that can be created. The first is a hard link and the other is a soft link (sometimes called symbolic link or symlink). The command to create these is the same - *ln*.

- **Hard Link** A hard link creates a second file that refers to the same file on the physical disk. This is achieved by having two filenames that point directly at the same file. This is normally used where file entries (links) are on the same filesystem. When a hard link is created then all the names that link to that file are given the same status. Deleting one of the files will break the link, but the file can still exist under the other linked filenames. This works by maintaining a counter of the number of filenames that the file has. When the number of filenames reaches zero then the file is considered to be deleted and is removed. The number of filenames for a file can be seen using the ‘*ls -l*’ command. The number following the file permissions indicates the number of linked filenames.

The following screenshot shows that filename1 and filename2 are to linked files with one of the file denoted by the 2 (in this case the same file, but they could be to completely different files), the file not link is a single file denoted by the 1.

```
ls -l
total 2432
-rw-r--r-- 2 stewart stewart 1241088 2009-01-23 15:26 filename1
-rw-r--r-- 2 stewart stewart 1241088 2009-01-23 15:26 filename2
-rw-r--r-- 1 stewart stewart 0 2009-01-23 15:26 not_link
du -h
1.2M
ln filename1 filename2
```

The default for the ln command is a hard link. Assuming filename1 already exists filename2 is created using: ln filename1 filename2

- **Soft Link** A soft link is sometimes referred to as a symbolic link or symlink. A filename created as a soft link is a special file that has the pathname of the file to redirect to. Behind the scenes when you try and access a symlink it just goes to the filename referred to instead.

In the following example a file has been created called original_file with a soft link to that same file called softlink_tofile. As you can see the ls command makes it clear that this is a link through the l at the beginning of the file permissions and due to the reference notation after the filename.

```
ls -l
total 440
-rw-r--r-- 1 stewart stewart 446464 2009-01-23 15:21 original_file
lrwxrwxrwx 1 stewart stewart 13 2009-01-23 15:20 softlink_tofile -> original_file
$ rm original_file
$ ls -l
total 0
lrwxrwxrwx 1 stewart stewart 13 2009-01-23 15:20 softlink_tofile -> original_file
$ cat softlink_tofile
cat: softlink_tofile: No such file or directory
The -s option is used on the ln command to create a softlink.
ln -s original_file softlink_tofile
```

Note that if original_file does not exist then the soft link will be created anyway. An attempt to read it will give the error message we encountered earlier, but an attempt to write to the file can create original_file.

- **Pitfalls while using links**
 - There are some things that you need to be aware of, particularly when using softlinks.
 - Some programs (e.g. Apache) can be configured to not follow soft links (from

a security

- When creating backup files you need to be aware of how the particular backup program / tool
- Using one method the program may not backup the file referenced resulting in an incomplete backup of the
- Using another method the program may end up backing up both as individual files using double the space for that
- Using the second method of the above could result in the file being restored as a real file rather than as a link, breaking their connection
- The original file could be deleted without realising that there are other sym-linked files referring to it

Despite these pitfalls there are many advantages to using both hard and soft links to provide multiple references to files.

2.4 Result/Observation

The linux commads for redirection of standard I/O, pipes, filters, job control and links in linux were studied and they were run on ArchLinux 4.10.11 and output was verified.

2.5 Viva

1. Which command redirects the standard output of command 1 to the standard input of command 2?
(A) `commandName > fileName`
(B) `command1 | command2`
(C) `commandName >> fileName`
(D) `sort < filename`
2. What is the difference between pipes and filters?
3. What is the command used to translate characters?
(A) `tr` (B) `grep` (C) `pr` (D) `tail`
4. Which command lists the jobs that the current shell is managing?
(A) `jobs` (B) `fg` (C) `ps` (D) `kill`
5. What is the command to temporarily become super user?
(A) `chmod` (B) `su` (C) `chown` (D) `chgrp`

3 Advanced Linux Commands

3.1 Aim

Advanced linux commands curl, wget, ftp, ssh and grep.

3.2 Overview

3.2.1 curl

curl is a tool to transfer data from or to a server, using one of the supported protocols (DICT, FILE, FTP, FTPS, GOPHER, HTTP, HTTPS, IMAP, IMAPS, LDAP, LDAPS, POP3, POP3S, RTMP, RTSP, SCP, SFTP, SMB, SMBS, SMTP, SMTPS, TELNET and TFTP). The command is designed to work without user interaction.

‘curl link’ - gives information about the website and outputs the html code.

‘curl -O link/file.html’ - copies the html code of the website to file.html

3.2.2 ssh

ssh (SSH client) is a program for logging into a remote machine and for executing commands on a remote machine. It is intended to replace rlogin and rsh, and provide secure encrypted communications between two untrusted hosts over an insecure network. X11 connections and arbitrary TCP/IP ports can also be forwarded over the secure channel.

ssh connects and logs into the specified hostname (with optional user name). The user must prove his/her identity to the remote machine using one of several methods depending on the protocol version used.

If command is specified, command is executed on the remote host instead of a login shell.

BASIC COMMAND

```
$ ssh remote_host
```

The remote host in this example is the IP address or domain name that you are trying to connect to. If your username is different on the remote system, you can specify it by using this syntax:

```
$ ssh remote_username@remote_host
```

TO COPY FILES

```
$ scp source destination
```

COPY A FILE FROM HOST TO LOCAL

```
$ scp localhostfile.txt
```

```
jsmith@remotehost.example.com:/home/jsmith/localhostfile.txt
```

COPY A FILE FROM LOCAL TO HOST

```
$ scp localhostfile.txt
```

```
jsmith@remotehost.example.com:/home/jsmith/localhostfile.txt
```

SHUTDOWN CONNECTED COMPUTER

```
$ ssh user@remote_computer
```

```
$ sudo poweroff
```

```
$ sudo rebootwline
```

3.2.3 wget

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

Wget is non-interactive, meaning that it can work in the background, while the user is not logged on. This allows you to start a retrieval and disconnect from the system, letting Wget finish the work. By contrast, most of the Web browsers require constant user's presence, which can be a great hindrance when transferring a lot of data.

Wget can follow links in HTML, XHTML, and CSS pages, to create local versions of remote web sites, fully recreating the directory structure of the original site. This is sometimes referred to as "recursive downloading." While doing that, Wget respects the Robot Exclusion Standard (/robots.txt). Wget can be instructed to convert the links in downloaded files to point at the local files, for offline viewing.

Wget has been designed for robustness over slow or unstable network connections; if a download fails due to a network problem, it will keep retrying until the whole file has been retrieved. If the server supports regetting, it will instruct the server to continue the download from where it left off.

```
wget http://www.openss7.org/repos/tarballs/strx25-0.9.2.1.tar.bz2
```

this command will download a single file and store it in the current repository.

```
ewget -O wget.zip http://ftp.gnu.org/gnu/wget/wget-1.5.3.tar.gz
```

Using -O (uppercase) option, downloads file with different file name. Here we have given wget.zip file name as show below.

```
wget -c http://www.openss7.org/repos/tarballs/strx25-0.9.2.1.tar.bz2
```

Restart a download which got stopped in the middle using wget -c option.

wget -mirror [WebsiteName] If you wish to retain a copy of any website that you may like to refer to/read locally, or maybe save a copy of your blog to the hard disk as back up, you may execute the wget command with mirror option.

```
wget -i download-file-list.txt
```

allows you to download multiple files stored in download-file-list.txt simultaneously.

3.2.4 ftp

The FTP (File Transfer Protocol) utility program is commonly used for copying files to and from other computers. These computers may be at the same site or at different sites thousands of miles apart. FTP is a general protocol that works on UNIX systems as well as a variety of other (non-UNIX) systems.

To connect your local machine to the remote machine, type

ftp machinename

where machinename is the full machine name of the remote machine, e.g., some-url.com.

If the name of the machine is unknown, you may type

ftp IP-Address

where machinenumber is the net address of the remote machine, e.g., 129.82.45.181.

In either case, this command is similar to logging onto the remote machine. If the remote machine has been reached successfully, FTP responds by asking for a loginname and password.

When you enter your own loginname and password for the remote machine, it returns the prompt

ftp >

and permits you access to your own home directory on the remote machine. You should be able to move around in your own directory and to copy files to and from your local machine using the FTP interface commands.

Table 4: FTP Interface Commands

?	request help or information about FTP Commands
cd	change directory in remote machine
close	terminate a connection with remote computer
delete	delete a file in remote computer
get	get a copy of a file in the remote machine to local machine
ls	list the names of files in the current directory in the remote machine
mkdir	make a new directory in the remote machine
pwd	print the working directory in remote machine
quit	exit ftp environment

3.2.5 grep

GREP : Global Regular Expression Print, Searches for text in a file, Can search for simple words. Can look for regular expression - more complex character strings(words followed by any no of spaces, followed by a digit or lowercase letter).

Searching the given string

`$ grep <literal string> ' filename` To search a specific string in a specified file
 Case insensitive search
`$ grep -i ?string? filename`
`$ grep -i ?string? FILE PATTERN`
 -This searches for given string/pattern case insensitively.
 Simple regular expressions
`?[0-9]?` look for any digit
`?[a-zA-Z]?` look for one upper or lower case letter
`?.` look for one charector
`?.*?` any number of charectors
`?n.?` a literal decimal point
`?n.161:?` dot, then 161, then colon
`?n.161[:]?` dot, then 161, then colon or space
 Advanced regular expressions
 Look for lines that hold either string1 or string2
`$ grep -E ?(string1—string2)? filename`
 Lines that have string1 followed by string2 on the same line, but possibly with other charectors in between.
`$ grep ?string1.*string2? filename`
 String1 has to be at the beginning of the line.
`$ grep ?string1? filename` Look for it at the end of the line. `$ grep ?string1$? filename`

3.3 Sample input/output

```

[santhisenan@arch ~]$ grep test test_file
grep: test_file: No such file or directory
[santhisenan@arch ~]$ grep test file_file
grep: file_file: No such file or directory
[santhisenan@arch ~]$ grep test file_test
test test
test
test
[santhisenan@arch ~]$ cat file_test
test test
test
fbmsdfk
oofkbn
test
[santhisenan@arch ~]$ _

```

Figure 2: Output 1

3.4 Result/Observation

The advanced linux commands ftp, ssh, wget, grep and curl were studied. The commands were run on ArchLinux 4.10.11.

3.5 Viva

1. What is the SSH command to copy files?
 - (A) \$ ssh remote_host
 - (B) \$ scp source destination
 - (C) \$ ssh remote_username@remote_host
 - (D) \$ sudo poweroff
2. What action does 'close' command do in a FTP environment?
 - (A) exit ftp environment
 - (B) delete a file in remote computer
 - (C) change directory in remote machine
 - (D) none of the above
3. What are 3 of the supporting protocols in CURL?
 - (A) UDP,POP3,TCP
 - (B) FTPS, GOPHER, HTTP.
 - (C) DTP,SCP,FTP
 - (D) SMB,SMBS,CACEP

4. What is the SSH command to shutdown connected computer?
- (A) `$ scp localhostfile.txt jsmith@remotehost.example.com:/home/jsmith/localhostfile.txt`
 - (B) `$ scp source destination`
 - (C) `$ ssh user@remote_computer`
 - (D) `$ scp localhostfile.txt jsmith@remotehost.example.com:/home/jsmith/localhostfile.txt`
5. Which of the following is an utility that does recursive downloading?
- (A) `grep`
 - (B) `ssh`
 - (C) `wget`
 - (D) `curl`

4 Shell programming

4.1 Aim

Write shell script to show various system configuration like

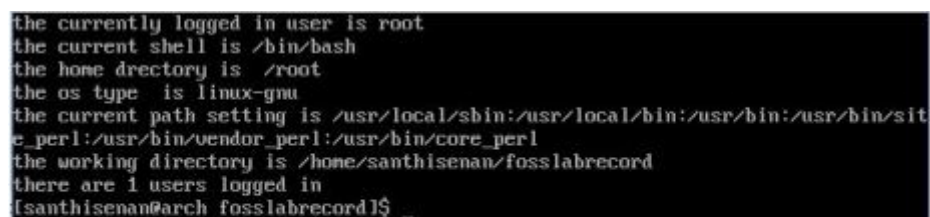
- Currently logged user and his login name
- Your current shell
- Your home directory
- Your operating system type
- Your current path setting
- Your current working directory
- Number of users currently logged in

4.2 Overview

4.2.1 Shellsript

```
1 clear
2 log=`who|wc -l`
3 echo "the currently logged in user is \${USER}"\newline
4 echo "the current shell is \${SHELL}"\newline
5 echo "the home drectory is \${HOME}"\newline
6 echo "the os type is \${OSTYPE}"\newline
7 echo "the current path setting is \${PATH}"\newline
8 echo "the working directory is \${PWD}"\newline
9 echo "there are \${log} users logged in"\newline
```

4.3 Sample input/output



```
the currently logged in user is root
the current shell is /bin/bash
the home drectory is /root
the os type is linux-gnu
the current path setting is /usr/local/sbin:/usr/local/bin:/usr/bin:/usr/bin/sit
e_perl:/usr/bin/vendor_perl:/usr/bin/core_perl
the working directory is /home/santhisenan/fossilabrecord
there are 1 users logged in
[santhisenan@arch fossilabrecord]$ _
```

Figure 3: Output

4.4 Result/Observations

The shell script for displaying various system configurations were made and the output was verified. The script was run on ArchLinux 4.10.11.

4.5 Viva

1. To know your home directory, what command will be use?
(A) echo "the current shell is \$SHELL"
(B) echo "the home drectory is "\$HOME"
(C) echo "the os type is \$OSTYPE"
(D) echo "the working directory is \$PWD"
2. What is the difference between System variable and user defined variable?
3. What is the use of echo command?
4. What is the use of 'echo "the current shell is \$SHELL"'?
(A) To know your home directory
(B) To know your current shell
(C) To know the OS type
(D) To know the current working directory
5. What is the numeric format of file permissions to read and execute?
(A) -rw-rw-rw-
(B) -r-r-r-
(C) -r-xr-xr-x
(D) -rwxr---

5 Shell script to show various system configurations

5.1 Aim

Write shell script to show various system configurations like

- your OS and version, release number, kernel version
- all available shells
- computer CPU information like processor type, speed etc
- memory information
- hard disk information like size of hard-disk, cache memory, model etc
- File system (Mounted)

5.2 Overview

Shell accept human readable commands from user and convert them into something which kernel can understand. It is a command language interpreter that execute commands read from input devices such as keyboards or from files.

5.3 Sample input/output

5.3.1 Shell Script

```
1 #!/bin/bash
2 echo -e "`cat /etc/os-release`"
3 echo -e "`cat /etc/shells`"
4 echo -e "`xset q`"
5 echo -e "`cat /proc/meminfo`"
6 echo -e "Driver: `sudo hdparm -I /dev/sda`"
7 echo -e "`cat /proc/mounts`"
```

5.4 Result/Observation

The shell script for displaying required system configurations was made and the output was verified. The shell script was run on ArchLinux 4.10.11.

5.5 Viva

1. Which command is used to display the operating system name
A) os B) unix C) kernel D) uname
2. Which command is used to display the unix version
A) uname -r
B) uname -n
C) uname -t
D) kernel

3. What is the use of "\$ uname -a" command?
4. Find / -name '*' will
- (A) List all files and directories recursively starting from /
 - (B) List a file named * in /
 - (C) List all files in / directory
 - (D) List all files and directories in / directory

6 Menu driven calculator

6.1 Aim

Write a shell script to implement a menu driven calculator with following functions

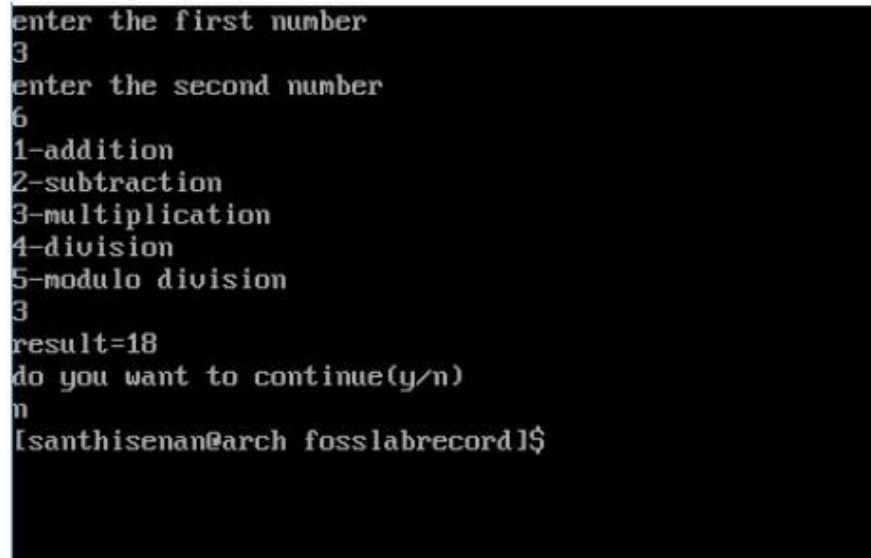
- Addition
- Subtraction
- Multiplication
- Division
- Modulus

6.2 Overview

6.2.1 Shell script

```
8 clear
9 i="y"
10 echo "enter the first number"
11 read n1
12 echo "enter the second number"
13 read n2
14 while [ $i = "y" ]
15 do
16 echo "1-addition"
17 echo "2-subtraction"
18 echo "3-multiplication"
19 echo "4-division"
20 echo "5-modulo division"
21 read c
22 case $c in
23 1)sum=`expr $n1 + $n2`
24 echo "result=$sum"
25 2)sum=`expr $n1 - $n2`
26 echo "result=$sum"
27 3)sum=`expr $n1 * $n2`
28 echo "result=$sum"
29 4)sum=`expr $n1 / $n2`
30 echo "result=$sum"
31 5)sum=`expr $n1 % $n2`
32 echo "result=$sum"
33 esac
34 echo "do you want to continue(y/n)"
35 read i
36 if [ $i != "y" ]
37 then
38 exit
39 fi
40 done
```


6.3 Sample input/output



```
enter the first number
3
enter the second number
6
1-addition
2-subtraction
3-multiplication
4-division
5-modulo division
3
result=18
do you want to continue(y/n)
n
[santhisenan@arch fosslabrecord]$
```

Figure 4: Output

6.4 Result/Observation

The shell script for a simple menu driven calculator was made and the output was verified. The script was run in ArchLinux 4.10.11 and screenshot of output is attached above.

6.5 Viva

1. What is the output of the following code

```
a=20
b=20
if [ $a == $b ]
then
If they are equal then print this
echo "a is equal to b"
else
else print this
echo "a is not equal to b"
fi
```

(A) a is not equal to b
(B) a is equal to b
(C) a is greater than b
(D) a is less than b

2. What is the general format of echo command?
(A) echo string (B) echo (string) (C) \$ echo string (D) \$ echo "string"
3. How to run a shell script?
(A) ./<fileName> (B) gcc filename (C) ./a.out (D) g++ filename "string"
4. What is the syntax of if statement?
- (A) if [expression]
then
statement
fi
- (B) if [expression]
then
statement1
else
statement2
fi
- (C) case in
Pattern 1) Statement 1;;
Pattern n) Statement n;;
esac
- (D) if ()
statement;
else
statements;

7 Script that accepts two arguments from the command line and operates on them

7.1 Aim

Write a script called `addnames` that is to be called as follows `./addnames ulist username`. Here `ulist` is the name of the file that contains list of user names and `username` is a particular student's username. The script should

- check that the correct number of arguments was received and print a message, in case the number of arguments is incorrect
- check whether the `ulist` file exists and print an error message if it does not
- check whether the username already exists in the file. If the username exists, print a message stating that the name already exists. Otherwise, add the username to the end of the list

7.2 Overview

```
1 if [[ $# -ne 2 ]]
2 then
3 echo "Invalid number of arguments"
4 exit
5 fi
6
7 if [[ ! (-a $1) ]]
8 then
9 echo "Not a valid file location or file doesn't exist"
10 exit
11 fi
12 NO=$(grep -c -e $2 $1)
13 if [[ $NO -eq 0 ]]
14 then
15 echo $2 >> $1
16 echo "Username is added"
17 exit
18 else
19 echo "Username already exists"
20 exit
21 fi
```

7.3 Sample input/output

```
[santhisenan@arch fosslabrecord1]$ ./addnames.sh
Enter the correct number of arguments : ./addnames.sh classlist
username
[santhisenan@arch fosslabrecord1]$ ./addnames.sh classlist santhisenan
The new claslist is
santhisenan
[santhisenan@arch fosslabrecord1]$ ./addnames.sh classlist santhisenan1
The new claslist is
santhisenan
santhisenan1
[santhisenan@arch fosslabrecord1]$ ./addnames.sh classlist santhisenan
Name already exists in the file
santhisenan
santhisenan1
[santhisenan@arch fosslabrecord1]$
```

Figure 5: Output

7.4 Result/Observation

The required shell script was made and the output was verified. The script was run on ArchLinux 4.10.11.

7.5 Viva

1. Shell script is preferable to other forms of programming because it
 - (A) Makes programming task easier
 - (B) Enhances portability
 - (C) Occupies less space
 - (D) All of these
2. Choose the incorrect statements.
 - (A) Shell scripts can accept arguments
 - (B) Shell scripts are interpreted
 - (C) Shell is a programming language
 - (D) Shell scripts are compiled
3. If 7 terminals are currently logged on. then the command `date ; who — wc -l`, displays
 - (A) date followed by 7
 - (B) date followed by 8
 - (C) date followed by 1
 - (D) an error message

8 Version Control System setup and usage using Git

8.1 Aim

Version Control System setup and usage using GIT. Try the following features.

- Creating a repository
- Checking out a repository
- Adding content to the repository
- Committing the data to a repository
- Updating the local copy
- Comparing different revisions
- Revert
- Conflicts and a conflict Resolution

8.2 Overview

- Git
Git is free and open source version control system, originally created by Linus Torvalds in 2005. Version control systems (VCS) are a category of software tools that help software teams maintain their source code. VCS allows developers to keep track of every modification made to the source code and also to turn back the clock and compare the earlier versions of code to fix their mistakes.
- Benefits of using a Version Control System
 - A complete long term history of every files.
 - Branching and merging
 - Traceability - being able to trace every change made in the software and connect it to project management and bug tracking softwares
- Creating a repository
You can use the UI provided by websites to create a repository and clone the repository to the local computer.
To clone an existing repository use `git clone https://example url.git` .
Use the command `git init` when inside your project's home folder to initialise an empty repository.
- Basic Git Commands

Table 5: Basic Commands

Command	Use
git init	create a new repository
git clone <repo>	clone the repository on to the local machine
git status	get the status of your local repository. It tells you how your project is progressing when compared to the remote repository
git add <filename>	tell git to start tracking the file
git add	add all files
git commit	commits all the added files. You must provide a commit message in the text editor that opens up
git commit -m"commit message"	a commit command with the message
git push origin master	save the committed changes to server
git pull -all	pull all changes from bitbucket server to your local repository

- Branching in Git

Branches are most powerful when you are working on a team. You can work on your part of the project by creating a branch and merging it to the main branch when you have finished. A branch provides an independent area for yourself to work.

There will be a main branch called master by default.

Table 6: Branching

Command	Use
git branch new_branch	create a new branch
git checkout new_branch	checkout to the new branch to start using it
git merge new_branch	merge the new branch to the master branch (perform this operation after checking out to the master branch)
	git branch -d branch name delete a branch

- Forking

You have only read access but not write access to another user's repository. This is where the concept of forking comes in. Here is the process of forking a repository

- Fork the repository to copy it to your account.
 - Clone the forked repository to your local computer
 - Make changes in the repository
 - Push the changes to your repository
 - Create a PULL REQUEST from the original repository you forked and add the changes you have made
 - Wait for the owner of the original repository to accept or reject changes
- To fork a repository use the website of the git client you are using.

- Undoing changes in a repository

- the git checkout command serves three distinct functions - checking out files , checking out commits , checking out branches

Checking out a commit makes the entire working directory match that commit. This can be used to view an old state of your project without altering the current state in any way.

How to use checkout

git log - oneline will show the ID of each commit made

use `git checkout <commit id>` to go to that commit

- `git revert <commit>` can also be used to undo changes.

This generates a new commit that undoes all of the changes introduced in `<commit>` and apply it to the current branch

- `git reset`

`git reset <file>` is used to remove the file from staging area but leave the working directory unchanged. This unstages a file without overwriting any changes

`git reset` is used to reset the staging area to match the most recent commit, but leave the working directory unchanged.

`git reset - - hard` is used to reset the staging area and the working directory to match the most recent commit. The `- - hard` tag tells git to overwrite all changes in the working directory.

`git reset <commit>` Move the current branch tip backward to `<commit>` , reset the staging area to match, but leave the working directory alone.

`git reset - - hard <commit>` moves the branch tip backwards to `<commit>` and resets both the staging area and working directory to match.

- Managing Conflicts

- When an user rebases or merges conflicts may occur. Conflicts occur when git cannot merge or rebase properly.
- If a merge conflict occurs, we have to resolve the conflict in order to move forward with the merge/rebase
- Run a `git status` to see where the problem is.

- Edit the file to resolve the conflict.
 - Add the files again and use git rebase - - continue
 - If you are not able to resolve the conflict, use git rebase - - abort to abort the rebase.(similarly abort the merge)
 - Use git push origin master to push the changes
- Comparing Different Revisions
git diff <commit> <commit> can be used to compare two different commits
here the <commit> is the commit id of the specific commit

8.3 Result/Observation

Basic git operations were familiarised. All the operations were performed in Git 2.10.1.

8.4 Viva

1. What is command to create a new repository?
(A) git add
(B) git status
(C) git init
(D) git clone<repo>
2. What does the command 'git commit -m"commit message"' is used for?
(A) tell git to start tracking the file
(B) a commit command with the message
(C) save the committed changes to server
(D) clone the repository on to the local machine
3. What does the command 'git push origin master' is used for?
(A) tell git to start tracking the file
(B) a commit command with the message
(C) save the committed changes to server
(D) clone the repository on to the local machine
4. What does the command 'git clone<repo>' is used for?
(A) tell git to start tracking the file
(B) a commit command with the message
(C) save the committed changes to server
(D) clone the repository on to the local machine
5. What does the command 'git add<filename>' is used for?
(A) tell git to start tracking the file
(B) a commit command with the message

- (C) save the committed changes to server
- (D) clone the repository on to the local machine

9 Shell script which starts on system boot up and kills every process which uses more than a specified amount of memory or CPU

9.1 Aim

Shell script which starts on system boot up and kills every process which uses more than a specified amount of memory or CPU.

9.2 Overview

```
41 #!/bin/sh
42 memlimit=10.0;
43 cpulimit=10.0;
44 while(true)
45 do
46 echo "script running.."
47 ps -e -o pmem=,cpu=,pid=,user=,comm= $|$ sort=-pmem $|$
48 while read size cpu pid user comm
49 do
50 kill_mem=0
51 kill_cpu=0
52 if [ "$user" = "santhisenan" ]
53 then
54 echo "Script Running..."
55 kill_mem=` echo "$size$>$$memlimit" $|$ bc `
56 kill_cpu=` echo "$cpu$>$$cpulimit" $|$ bc `
57 if [ "$kill_mem = 1" ]
58 then
59 echo "process with PID $pid killed"
60 kill $pid
61 elif [ "$kill_cpu = 1" ]
62 then
63 echo "process with PID $pid killed"
64 kill $pid
65 else
66 continue
67 fi
68 fi
69 done
70 sleep 1
71 done
```

```
[santhisenan@arch fosslabrecord1]$ ./killprocess.sh
script running..
script running..
script running..
script running..
script running..
```

Figure 6: Output

9.3 Sample input/output

9.4 Result/Observation

A shell script for killing processes that consume more than a specific amount of memory and cpu was made and the output was verified. The shell script was run on ArchLinux 4.10.11.

9.5 Viva

1. The command 'umask -S'
 - (A) prints the current mask using symbolic notation
 - (B) prints the current mask using octal numbers
 - (C) sets the mask to 000
 - (D) sets the mask to 777
2. Which option of the kill command sends the given signal name to the specified process?
 - (A) -l
 - (B) -n
 - (C) -s
 - (D) -a
3. Which command wait for the specified process to complete and return the exit status?
 - (A) sleep
 - (B) wait
 - (C) delay
 - (D) stop
4. Which command prints the accumulated user and system times for processes run from the shell?
 - (A) time
 - (B) times
 - (C) both time and times
 - (D) none of the mentioned

10 Introduction to packet management system

10.1 Aim

Given a set of RPM or DEB, build and maintain, and serve packages over http or ftp. Configure client systems to access the package repository.

10.2 Overview

Most modern Unix-like operating systems offer a centralized mechanism for finding and installing software. Software is usually distributed in the form of packages, kept in repositories. Working with packages is known as package management. Packages provide the basic components of an operating system, along with shared libraries, applications, services, and documentation. A package management system does much more than one-time installation of software. It also provides tools for upgrading already-installed packages. Package repositories help to ensure that code has been vetted for use on your system, and that the installed versions of software have been approved by developers and package maintainers.

When configuring servers or development environments, it's often necessary look beyond official repositories. Packages in the stable release of a distribution may be out of date, especially where new or rapidly-changing software is concerned. Nevertheless, package management is a vital skill for system administrators and developers, and the wealth of packaged software for major distributions is a tremendous resource. This guide is intended as a quick reference for the fundamentals of finding, installing, and upgrading packages on a variety of distributions, and should help you translate that knowledge between systems.

Package Management Systems: A Brief Overview Most package systems are built around collections of package files. A package file is usually an archive which contains compiled binaries and other resources making up the software, along with installation scripts. Packages also contain valuable metadata, including their dependencies, a list of other packages required to install and run them. Most modern Unix-like operating systems offer a centralized mechanism for finding and installing software. Software is usually distributed in the form of packages, kept in repositories. Working with packages is known as package management. Packages provide the basic components of an operating system, along with shared libraries, applications, services, and documentation. A package management system does much more than one-time installation of software. It also provides tools for upgrading already-installed packages. Package repositories help to ensure that code has been vetted for use on your system, and that the installed versions of software have been approved by developers and package maintainers. When configuring servers or development environments, it's often necessary look beyond official repositories. Packages in the stable release of a distribution may be out of date, especially where new or rapidly-changing software is concerned. Nevertheless, package management is a vital skill for system administrators and developers, and the wealth of packaged software for major distributions is a tremendous resource.

This guide is intended as a quick reference for the fundamentals of finding, installing, and upgrading packages on a variety of distributions, and should help you translate that knowledge between systems. **Package Management Systems: A Brief Overview**

Most package systems are built around collections of package files. A package file is usually an archive which contains compiled binaries and other resources making up the software, along with installation scripts. Packages also contain valuable metadata, including their dependencies, a list of other packages required to install and run them.

Operating System	Format	Tools(s)
Debian	.deb	apt apt-cache, apt-get,dpkg
Ubuntu	.deb	apt apt-cache , apt-get , dpkg
Fedora	.rpm	dnf

In Debian and systems based on it, like Ubuntu, Linux Mint, and Raspbian, the package format is the .deb file. APT, the Advanced Packaging Tool, provides commands used for most common operations: Searching repositories, installing collections of packages and their dependencies, and managing upgrades.

APT commands operate as a front-end to the lower-level dpkg utility, which handles the installation of individual .deb files on the local system, and is sometimes invoked directly. Recent releases of most Debian-derived distributions include the apt command, which offers a concise and unified interface to common operations that have traditionally been handled by the more-specific apt-get and apt-cache . Its use is optional, but may simplify some tasks.

CentOS, Fedora, and other members of the Red Hat family use RPM files. In CentOS, yum is used to interact with both individual package files and repositories. In recent versions of Fedora, yum has been supplanted by dnf , a modernized fork which retains most of yum 's interface. Update Package Lists Most systems keep a local database of the packages available from remote repositories. It's best to update this database before installing or upgrading packages.

As a partial exception to this pattern, yum and dnf will check for updates before performing some operations, but you can ask them at any time whether updates are available.

Upgrade Installed Packages Making sure that all of the installed software on a machine stays up to date would be an enormous undertaking without a package system. You would have to track upstream changes and security alerts for hundreds of different packages. While a package manager doesn't solve every problem you'll encounter when upgrading software, it does enable you to maintain most system components with a few commands. On FreeBSD, upgrading installed ports can introduce breaking changes or require manual configuration steps. It's best to read /usr/ports/UPDATING before upgrading with portmaster .

Find a Package

Most distributions offer a graphical or menu-driven front end to package collections. These can be a good way to browse by category and discover new software. Often, however, the quickest and most effective way to locate a package is to search with command-line tools.

View Info About a Specific Package

When deciding what to install, it's often helpful to read detailed descriptions of packages. Along with human-readable text, these often include metadata like version numbers and a list of the package's dependencies.

10.3 Result/Observation

Familiarised with packet management system.

10.4 Viva

1. Packet management system tool used in CentOS OS?
(A)yum (B)dnf (C)make (D)pkg
2. what is format used in Ubuntu operating system?
(A)rpm (B)ports (C)deb (D)txz
3. what is the command for update package lists in Fedora?
(A)dnf check-update
(B)yum check-update
(C)sudo apt update
(D)sudo apt-get update
4. which system uses the command sudo apt update?
(A)Ubuntu (B)CentOS (C)Fedora (D)FreeBSD

11 Simple Text Processing using Perl and Awk

11.1 Aim

Perform simple text processing using Perl, Awk.

11.2 Overview

11.2.1 Perl

- Match Operator

```
1 #!/usr/bin/perl
2 \ $string = "Hello World";
3 if($string =~ /match/){
4 print "Match found";
5 }
6 else{
7 print "It is not matching "
8 }
9 $string = "matchstick";
10 if($string =~ /match/){
11 print "Match found";
12 }
13 else{
14 print "It is not matching";
15 }
```

Output : It is not matching
Match found

- Substitution operator

```
16 #!/usr/bin/perl
17 $string = "The cat sat on the mat";
18 $string =~ s/cat/dog/;
19 print $string;
```

Output :
The Dog sat on the ground

11.2.2 Awk

- file data.txt
 - 1) Ashok Physics 80
 - 2) Rahul Maths 90
 - 3) Shyam Biology 87
 - 4) Kedar English 85

5) Hari History 89

- awk command file - cmd.awk
Print all the data on data file
print
Print only certain columns
print \$3 " " \$4

11.3 Result/Observation

The script for simple word processing was made and the output was verified. The script was run using perl 5, version 24, subversion 1 (v5.24.1) and GNU Awk 4.1.4 was used.

11.4 Viva

1. Which one of the following is the most powerful filter?
a) awk b) grep c) sed d) perl
2. A perl program runs in a special interpretive mode.
a) True
b) False
3. It is often more convenient to save perl program files with ____ extension.
a) .gp b) .sh c) .awk d) .pl
4. perl variables have no type and no initialization.
a) True
b) False
5. Which function is used by perl for displaying the length of a string?
a) string b) len c) split d) length

12 Running PHP

12.1 Aim

Running PHP : simple applications like login forms after setting up a LAMP stack.

12.2 Overview

- **Install Apache**
To install Apache you must install the Metapackage apache2. This can be done by searching for and installing in the Software Centre, or by running the following command.
`sudo apt-get install apache2`
- **Install MySQL**
To install MySQL you must install the Metapackage mysql-server. This can be done by searching for and installing in the Software Centre, or by running the following command.
`sudo apt-get install mysql-server`
- **Install PHP**
To install PHP you must install the Metapackages php5 and libapache2-mod-php5. This can be done by searching for and installing in the Software Centre, or by running the following command.
`sudo apt-get install php5 libapache2-mod-php5`
- **Restart Server**
Your server should restart Apache automatically after the installation of both MySQL and PHP. If it doesn't, execute this command.
`sudo /etc/init.d/apache2 restart`
- **Check Apache**
Open a web browser and navigate to `http://localhost/`. You should see a message saying It works!
`sudo apt-get install mysql-server`
- **Install MySQL**
To install MySQL you must install the Metapackage mysql-server. This can be done by searching for and installing in the Software Centre, or by running the following command.
`sudo apt-get install mysql-server`
- **Check Php**
You can check your PHP by executing any PHP file from within `/var/www/`. Alternatively you can execute the following command, which will make PHP run the code without the need for creating a file .
`php -r 'echo "Your PHP installation is working fine.";'`

12.3 Sample input/output

12.3.1 Php - Example

- config.php

```
<?php
define( DB_SERVER , localhost );
define( DB_USERNAME , fc );
define( DB_PASSWORD , );
define( DB_DATABASE , fossdb );
$db = mysqli_connect(DB_SERVER,DB_USERNAME,DB_PASSWORD,
    DB_DATABASE);
?>
```

- login.php

```
<?php
include( config . php );
session_start();
$error="";
if($_SERVER["REQUEST_METHOD"] == "POST"){
    $myusername = $_POST[ username ];
    $mypassword = $_POST[ password ];
    $sql = "SELECT * FROM admin WHERE username = $myusername
        and password = $mypassword ";
    $result = mysqli_query($db,$sql) or die(mysqli_error($db));
    $row = mysqli_fetch_array($result,MYSQLI_ASSOC);
    $count = mysqli_num_rows($result);
    if($count == 1) {
        session_register("myusername");
        $_SESSION[ username ] = $myusername;
        $_SESSION[ loggedin ] = true;
        // header( Location : http://localhost/login/welcome.php );
        // exit;
        error_reporting(E_ALL | E_WARNING | E_NOTICE);
        ini_set( display_errors , TRUE);
        flush();
        header("Location:http://localhost/login/welcome.php");
        die( should have redirected by now );
    }
    else {
        $error = "Your Login Name or Password is invalid";
    }
}

?>
<html>
<head>
<title>Login Page$</title>
</head>
```

```

<div><?php echo error ?></div>
<body>
<form action="login.php" method="POST">
<input type="text" placeholder="username" name="username">
<input type="password" placeholder="password" name="password">
<input type="submit" value="submit">
</form>
</body>
</html>

```

- session.php

```

<?php
include( config . php );
session_start();
$user_check = $_SESSION[ username ];
$ses_sql = mysqli_query($db,"select username from admin where
    username = $user_check ");
$row = mysqli_fetch_array($ses_sql,MYSQLI_ASSOC);
$login_session = $row[ username ];
if(!isset($_SESSION[ login_user ])){
header("location:login.php");
}
?>

```

- welcome.php

```

<html>
<body>
Logged in
</body>
</html>

```

12.4 Result/Observation

LAMP stack was successfully installed and a sample login form using PHP and mysql was made. PHP 5.6 and mySQL 5.7 were used to run the code on Ubuntu 15.10.

12.5 Viva

1. What do the initials of PHP stand for?
 - (A) Personal homepage
 - (B) Pretext hypertext
 - (C) private hyper texter
 - (D) Hypertext Preprocessor
2. PHP is an example of " " scripting language ?
 - (A) Server-side

- (B) Client-side
- (C) Browser-side
- (D) In-side

3. Which of the following is not true?
- (A) PHP can be used to develop web applications
 - (B) PHP makes a website dynamic
 - (C) PHP applications can not be compile
 - (D) PHP can not be embedded into html.

13 Virtualization Environment

13.1 Aim

To test an applications, new kernels and isolate applications.

13.2 Overview

13.2.1 Algorithm

- Step 1. Open VirtualBox and select New . A new window will come out.
- Step 2. Choose your guest OS and architecture (32 vs. 64 bit, e.g select Ubuntu)
- Step 3. Set your Base Memory (RAM)
- Step 4. Click next until it show the vm storage size. Put how much space you need depending on your hardisk and finish the wizard by clicking the create button.
- Step 5. On VirtualBox main window, select START and pick your MEDIA SOURCE. In your case, select iso on your desktop.
- Step 6. Finish the installation as normal install.
- Step 7. Remove your installation .iso from the virtual optical disk drive before restarting the VM.
- Step 8. Install Guest Additions .

Open Virtualbox and click at New button.

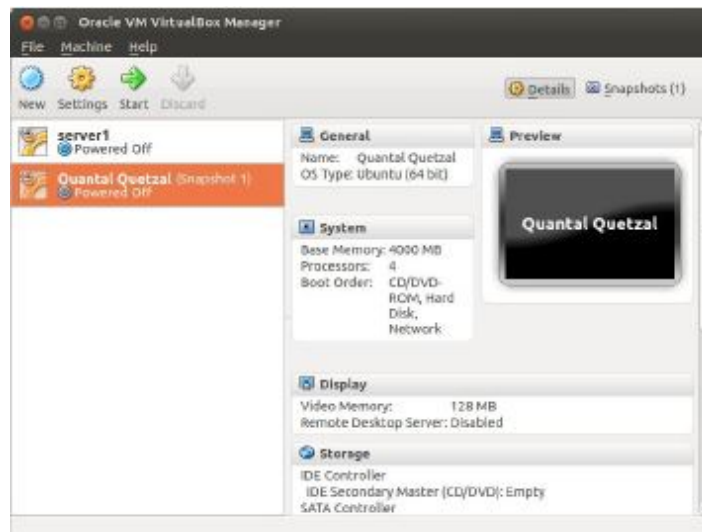


Figure 7: Setup Wizard will appear and click at Next button.

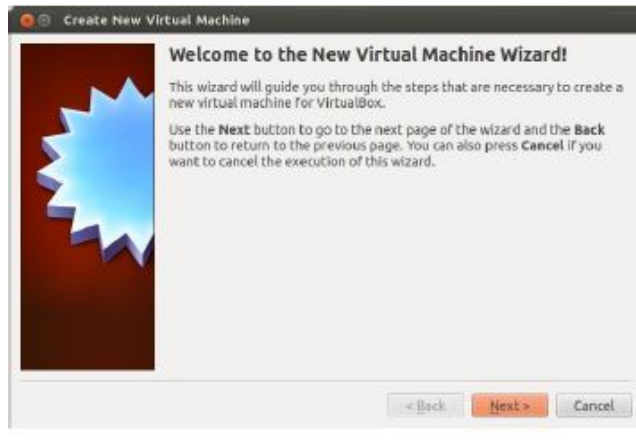


Figure 8: Enter your Virtual Machine name, and choose your guest OS and architecture (32- vs. 64-bit) from the dropdown menu and click Next button. A 64-bit guest needs the CPU virtualization technology (VT-x AMD/V) to be enabled in BIOS.



Figure 9: Enter memory (RAM) to reserve for your virtual machine and click Next button. Leave enough memory to the host OS.

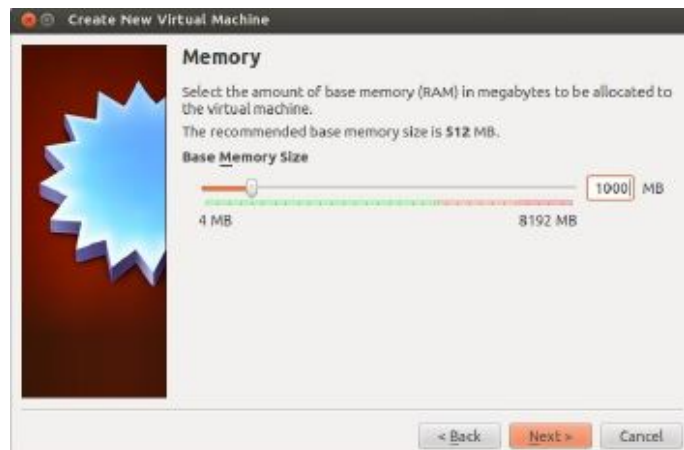


Figure 10: Tick at Startup Disk and Create New Hard disk and click at Next button.



Figure 11: Choose the type of file that you want to use for virtual disk and click Next button.



Figure 12: Choose your storage detail and click Next button.



Figure 13: Enter the size of your virtual disk (in MB) and click Next button. A dynamically growing virtual disk will only use the amount of physical hard drive space it needs. It is better to be rather generous to avoid running out of guest hard drive space.



Figure 14: You will see the detail of your input here. Click Create button to continue.

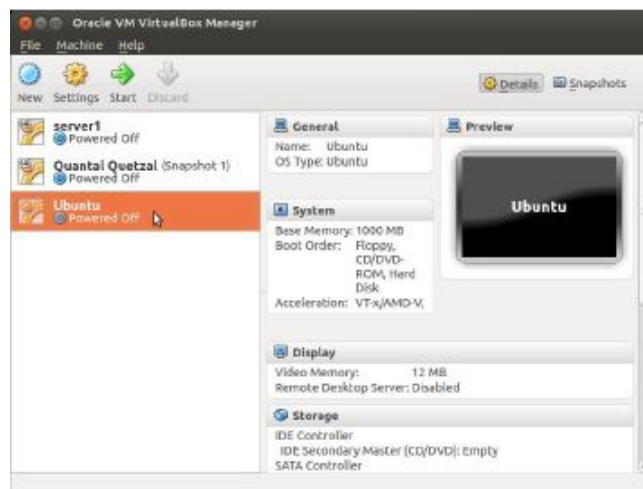


Figure 15: "First Run Wizard" will appear and click Next button.



Figure 16: Click at 'folder' icon and choose your Ubuntu iso directory.



Figure 17: In 'Summary' box, click Start button

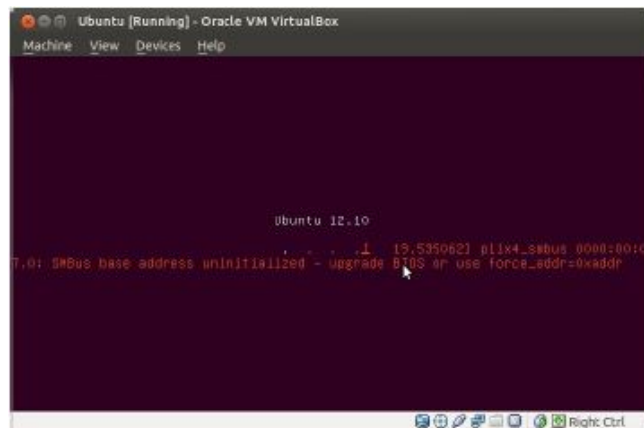
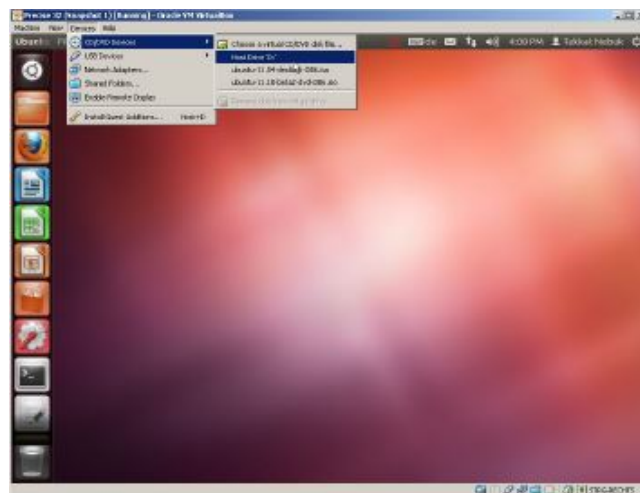


Figure 18: After a successful installation we have to remove our installation .iso image from the virtual optical drive before we reboot. This can be done from the " Devices" menu or by removing the .iso from the VM settings



13.3 Result/Observation

Thus the program for virtualization is performed successfully.

13.4 Viva

1. The communication between two ESXi hosts is called FT logging
 (A) True
 (B) False
2. What is VVol?
 (A) virtual volume (B) viral volume (C) viral velocity (D) virtual velocity

3. A hypervisor is a program that enables multiple operating systems to share a single hardware host.
- (A) True
 - (B) False

14 Compiling From The Source

14.1 Aim

Learn about the various build systems used like the auto* family, cmake, ant etc. instead of just running the commands. This could involve the full process like fetching from a cvs and also include autoconf, automake etc.,

14.2 Overview

Open source software is distributed in source code form. In case of popular software Linux distributions will often have the software packaged in their repositories. If the package is not in the repository the user has to compile the software from source. To do this the user has to understand about the build system used in the project.

The GNU build system, also known as the Auto tools, is a suite of programming tools designed to assist in making source-code packages portable to many Unix-like systems. It can be difficult to make a software program portable: the C compiler differs from system to system; certain library functions are missing on some systems; header files may have different names. One way to handle this is write conditional code, with code blocks selected by means of preprocessor directives (ifdef); but because of the wide variety of build environments this approach quickly becomes unmanageable. The GNU build system is designed to address this problem more manageably.

14.2.1 Tools included in the GNU build system

The GNU build system comprises the GNU utility programs Autoconf, Automake, and Libtool. Other related tools frequently used with the GNU build system are GNUs make program, GNU gettext, pkgconfig, and the GNU Compiler Collection, also called GCC.

14.2.2 GNU Autoconf

Autoconf generates a configure script based on the contents of a configure.ac file which characterizes a particular body of source code. The configure script, when run, scans the build environment and generates a subordinate config.status script which, in turn, converts other input files and most commonly Makefile.in into output files (Makefile) which are appropriate for that build environment. Finally the make program uses Makefile to generate executable programs from source code.

The complexity of the GNU build system reflects the variety of circumstances under which a body of source code may be built. If a source code file is changed then it suffices to re-run make which only recompiles that part of the body of the source code affected by the change.

If a .in file has changed then it suffices to re-run config.status and make. If the body of source code is copied to another computer then it suffices to re-run configure (which

runs `config.status`) and `make`. (For this reason source code using the GNU build system is normally distributed without the files that `configure` generates.)

If the body of source code is changed more fundamentally then `configure.ac` and the `.in` files need to be changed and all subsequent steps also followed.

To process files, `autoconf` uses the GNU implementation of the `m4` macro system. `Autoconf` comes with several auxiliary programs such as `Autoheader`, which is used to help manage C header files; `Autoscan`, which can create an initial input file for `Autoconf`; and `ifnames`, which can list C preprocessor identifiers used in the program.

14.2.3 GNU Automake

`Automake` helps to create portable `Makefiles`, which are in turn processed with the `make` utility. It takes its input as `Makefile.am`, and turns it into `Makefile.in`, which is used by the `configure` script to generate the file `Makefile` output.

14.2.4 GNU Libtool

`Libtool` helps manage the creation of static and dynamic libraries on various Unix-like operating systems. `Libtool` accomplishes this by abstracting the library-creation process, hiding differences between various systems (e.g. GNU/Linux systems vs. Solaris).

14.2.5 Gnulib

`Gnulib` simplifies the process of making software that uses `Autoconf` and `Automake` portable to a wide range of systems.

14.2.6 Make

In software development, `make` is a utility that automatically builds executable programs and libraries from source code by reading files called `makefiles` which specify how to derive the target program. `Make` can decide where to start through topological sorting. Though integrated development environments and language-specific compiler features can also be used to manage the build process in modern systems, `make` remains widely used, especially in Unix.

`Make` is typically used to build executable programs and libraries from source code. Generally hough, any process that involves transforming a dependency file to a target result (by executing some number of arbitrary commands) is applicable to `make`. To cite an example, `make` could be used to detect a change made to an image file (the dependency) and the target actions that result might be to convert the file to some specific format, copy the result into a content management system, and then send email to a predefined set of users that the above actions were performed.

14.2.7 Cmake

CMake is a unified, cross-platform, open-source build system that enables developers to build, test and package software by specifying build parameters in simple, portable text files. It works in a compiler independent manner and the build process works in conjunction with native build environments, such as make, Apple's Xcode and Microsoft Visual Studio. It also has minimal dependencies, C++ only.

CMake is open source software. CMake can:

- Create libraries
- Generate wrappers
- Compile source code
- Build executable in arbitrary combination

14.2.8 Apache Ant

Apache Ant is a software tool for automating software build processes. It is similar to Make but is implemented using the Java language, requires the Java platform, and is best suited to building Javaprojects. The most immediately noticeable difference between Ant and Make is that Ant uses XML to describe the build process and its dependencies, whereas Make uses Makefile format. By default the XML file is named build.xml. Ant is an Apache project. It is open source software, and is released under the Apache Software License.

Pre-requisites:

To ensure that all tools required are installed.

Type the following commands in terminal and type the password for root user, when prompted.

Step:1

```
[fossilab@fossilab]$ su
```

```
Password:(admin123)
```

Step 2:

```
[root@ fossilab fossilab] rpm -qa cmake
```

```
cmake-2.8.2-2.fc14.i686
```

Step:3

```
fossilab fossilab] rpm -qa ant
```

```
ant-1.7.1-13.fc13.i686
```

Step:4

```
[root@ fossilab fossilab] rpm -qa java-1.6.0-openjdk-devel
```

```
java-1.6.0-openjdk-devel-1.6.0.0-44.1.9.1.fc14.i686
```

Step:5

```
[root@ fossilab fossilab] exit
```

Exercises:

Create a directory for all the programs in the exercise.

Step:6

```
[fossilab@fossilab ~]$ mkdir build_systems
```

Step:7

```
[fossilab@fossilab ~]$ cd build_systems
```

1. Make

We shall be using a simple program written in C and write a makefile to compile the program.

Step:8

```
[fossilab@fossilab build_systems]$ mkdir gnumake
```

Step:9

```
[fossilab@fossilab build_systems]$ cd gnumake
```

Step:10

```
[fossilab@fossilab gnumake]$ gedit squareroot.c
```

Type and save the following simple program for square root of a number

```
// A simple program that computes the square root of a number
```

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
#include <math.h>
```

```
int main (int argc, char *argv[])
```

```
{
```

```
    if (argc < 2)
```

```
    {
```

```
        fprintf(stdout,"Usage: %s number\n",argv[0]);
```

```
        return 1;
```

```
    }
```

```
    double inputValue = atof(argv[1]);
```

```
    double outputValue = sqrt(inputValue);
```

```
    fprintf(stdout,"The square root of %g is %g",inputValue, outputValue);
```

```
    return 0;
```

Close gedit and test it by compiling it once:

Step:11

```
[fossilab@fossilab gnumake]$ gcc squareroot.c -o squareroot -lm
```

Step:12

```
[fossilab@fossilab gnumake]$ ./squareroot 25
```

```
The square root of 25 is 5
```

Write a simple makefile to compile the program.

Step:13

```
[fossilab@fossilab gnumake]$ gedit Makefile
```

Type and save the following code

Commands start with TAB not spaces

```
CC= gcc
```



```

CFLAGS= -g
LDFLAGS = -lm
all: squareroot
squareroot: squareroot.o
squareroot.o: squareroot.c
clean:
rm -f squareroot squareroot.o
Close gedit and test the Makefile
Step:14
[fossl@fossl gnumake]$ make
make: Nothing to be done for 'all'.
Step:15
[fossl@fossl gnumake]$ make clean
Step:16
[fossl@fossl gnumake]$ make
gcc -g -c -o squareroot.o squareroot.c
gcc -lm squareroot.o -o squareroot
Step:17
[fossl@fossl gnumake] ./squareroot 25
The square root of 25 is 5
Close gedit and test the Makefile
Step:18
[fossl@fossl gnumake]$ make
make: Nothing to be done for 'all'.
Step:19
[fossl@fossl gnumake]$ make clean
Step:20
[fossl@fossl gnumake]$ make
gcc -g -c -o squareroot.o squareroot.c
gcc -lm squareroot.o -o squareroot
Step:21
[fossl@fossl gnumake] ./squareroot 25
The square root of 25 is 5

```

2. Cmake

Write a simple script for CMake to compile the previously written program. Create a new directory and copy the source code to it.

```

Step:22
[fossl@fossl gnumake] mkdir cmake
Step:23
[fossl@fossl gnumake] cp squareroot.c
/home/fossl/build_systems/gnumake/cmake/
Create configuration files for Cmake.
Step:24

```

```
[fossilab@fossilab gnumake] cd cmake
```

Step:25

```
[fossilab@fossilab cmake]$ gedit CmakeLists.txt
```

Type and save the following code

```
cmake_minimum_required (VERSION 2.6)
```

```
project (squareroot)
```

```
add_executable(squareroot squareroot.c)
```

```
TARGET_LINK_LIBRARIES(squareroot m)
```

CMake is commonly use with out of source builds ie, we build the program in a directoryseparate from the source. We use the generated makefile to compile the program.

Step:26

```
[fossilab@fossilab cmake]$ mkdir build
```

Step:27

```
[fossilab@fossilab cmake]$ cd build
```

Step:28

```
[fossilab@fossilab build]$ cmake ..
```

Step:29

```
[fossilab@fossilab build]$ make
```

Step:30

```
[fossilab@fossilab build]$ ./squareroot 25
```

The square root of 25 is 5

3. Apache Ant

Create a new directory for the ant exercise. (open a nee terminal)

Step:31

```
[fossilab@fossilab ]$ cd build_systems
```

Step:32

```
[fossilab@fossilab build_systems]$ mkdir ant
```

Step:33

```
[fossilab@fossilab build_systems]$ cd ant
```

Step:34

```
[fossilab@fossilab ant]$ mkdir -p src/hello
```

Step:35

```
[fossilab@fossilab ant]$ gedit src/hello/HelloWorld.java
```

Type and save the following code

```
package hello;
```

```
public class HelloWorld
```

```
public static void main(String[ ] args)
```

```
System.out.println("Hello World");
```

Step:36

```
[fossilab@fossilab ant]$ mkdir -p build/classes
```

Step:37

```
[fossilab@fossilab ant]$ javac -sourcepath src -d build/classes/ src/hello/HelloWorld.java
```

Step:38

```
[fossilab@fossilab ant]$ java -cp build/classes hello.HelloWorld
```

Hello World

Step:39

```
[fossilab@fossilab ant]$
```

Write the ant build script.

```
[fossilab@fossilab ant]$ gedit build.xml
```

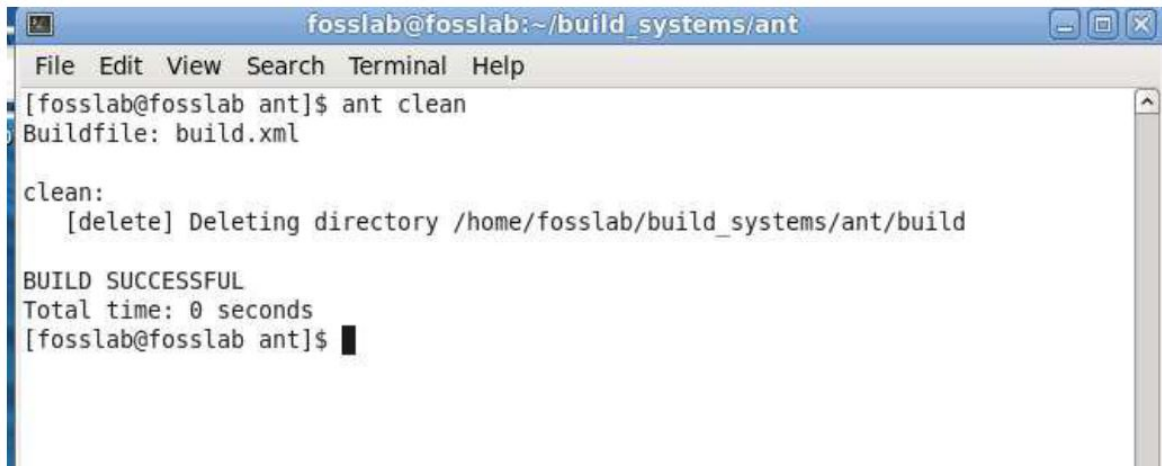
Type the following code and save

```
<project>
<target name="clean">
<delete dir="build" />
</target>
<target name="compile">
<mkdir dir="build/classes" />
<javac srcdir="src" destdir="build/classes" />
</target>
<target name="jar">
<mkdir dir="build/jar" />
<jar destfile="build/jar/HelloWorld.jar" basedir="build/classes">
<manifest>
<attribute name="Main-Class" value="hello.HelloWorld" />
</manifest>
</jar>
</target>
<target name="run">
<java jar="build/jar/HelloWorld.jar" fork="true" />
</target>
</project>
```

Now the project can be compile and run using ant.

Step:40

```
[fossilab@fossilab ant]$ ant clean
```

A terminal window titled 'fossilab@fossilab:~/build_systems/ant' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal shows the command '[fossilab@fossilab ant]\$ ant clean' and its output: 'Buildfile: build.xml', 'clean:', '[delete] Deleting directory /home/fossilab/build_systems/ant/build', 'BUILD SUCCESSFUL', 'Total time: 0 seconds', and '[fossilab@fossilab ant]\$' with a cursor.

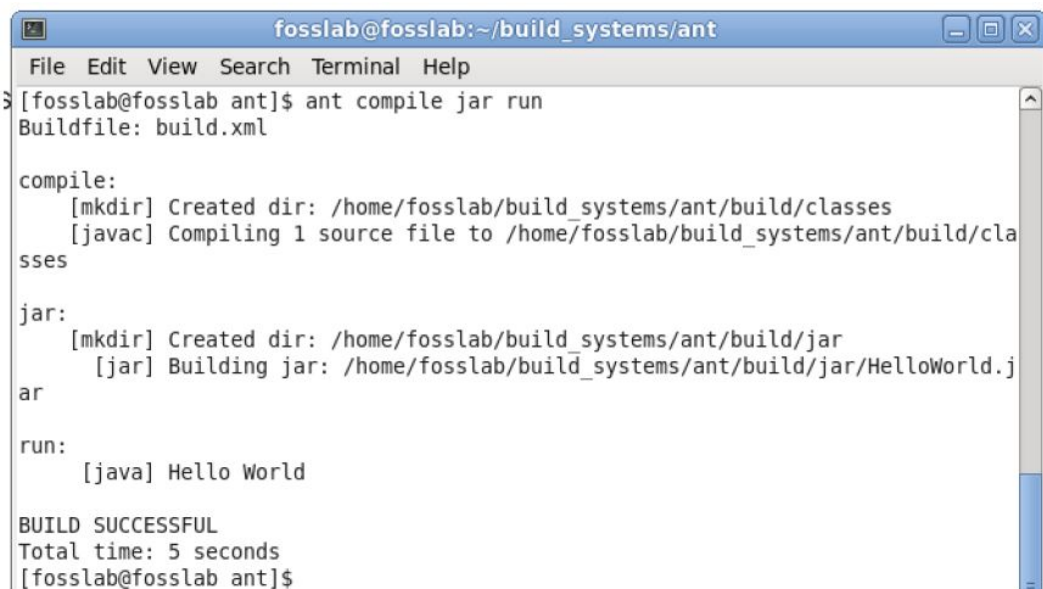
```
fossilab@fossilab:~/build_systems/ant
File Edit View Search Terminal Help
[fossilab@fossilab ant]$ ant clean
Buildfile: build.xml

clean:
  [delete] Deleting directory /home/fossilab/build_systems/ant/build

BUILD SUCCESSFUL
Total time: 0 seconds
[fossilab@fossilab ant]$
```

Step:41

[fossilab@fossilab ant]\$ ant compile jar run

A terminal window titled 'fossilab@fossilab:~/build_systems/ant' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal shows the command '[fossilab@fossilab ant]\$ ant compile jar run' and its output: 'Buildfile: build.xml', 'compile:', '[mkdir] Created dir: /home/fossilab/build_systems/ant/build/classes', '[javac] Compiling 1 source file to /home/fossilab/build_systems/ant/build/classes', 'jar:', '[mkdir] Created dir: /home/fossilab/build_systems/ant/build/jar', '[jar] Building jar: /home/fossilab/build_systems/ant/build/jar/HelloWorld.jar', 'run:', '[java] Hello World', 'BUILD SUCCESSFUL', 'Total time: 5 seconds', and '[fossilab@fossilab ant]\$' with a cursor.

```
fossilab@fossilab:~/build_systems/ant
File Edit View Search Terminal Help
[fossilab@fossilab ant]$ ant compile jar run
Buildfile: build.xml

compile:
  [mkdir] Created dir: /home/fossilab/build_systems/ant/build/classes
  [javac] Compiling 1 source file to /home/fossilab/build_systems/ant/build/classes

jar:
  [mkdir] Created dir: /home/fossilab/build_systems/ant/build/jar
  [jar] Building jar: /home/fossilab/build_systems/ant/build/jar/HelloWorld.jar

run:
  [java] Hello World

BUILD SUCCESSFUL
Total time: 5 seconds
[fossilab@fossilab ant]$
```

GNU Autotools

Copy the file hello-2.7.tar.gz to the buildsystems project directory and uncompress it

Step:42

[fossilab@fossilab ~]\$ cd Downloads

Step:43

[fossilab@fossilab Downloads]\$ mv hello-2.7.tar.gz /home/fossilab/build_systems

Step:44

[fossilab@fossilab Downloads]\$ cd /home/fossilab/build_systems

Step:45

```
[fossilab@fossilab build_systems]$ tar -xzf hello-2.7.tar.gz
```

Step:46

```
[fossilab@fossilab build_systems]$ cd hello-2.7
```

Step:47

```
[fossilab@fossilab hello-2.7]$
```

Step:48

```
[fossilab@fossilab hello-2.7]$ ./configure
```

Step:49

```
[fossilab@fossilab hello-2.7]$ make
```

Step:49

```
[fossilab@fossilab hello-2.7]$ src/hello
```

The program will now reside in the src directory. To install the program log in as root.

Step:50

```
[fossilab@fossilab hello-2.7]$ su
```

```
Password:(admin123)
```

Step:51

```
[root@fossilab hello-2.7] make install
```

Step:52

```
[root@fossilab hello-2.7] exit
```

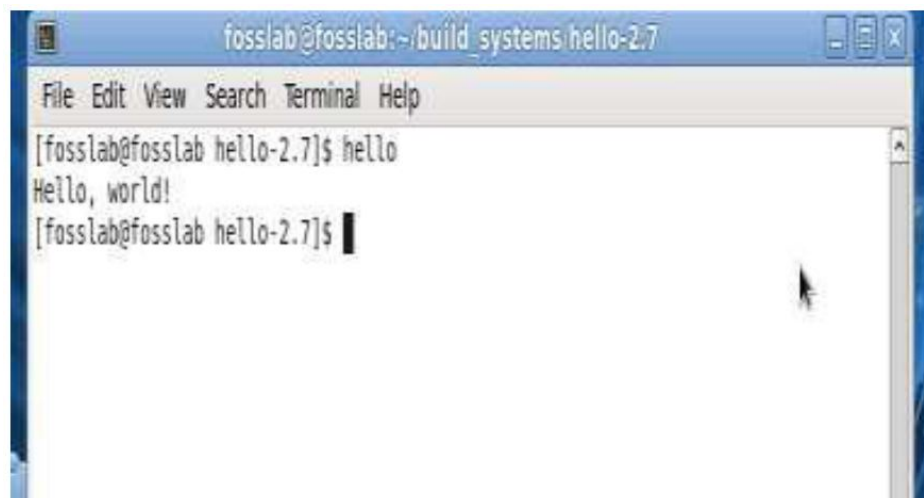
```
exit
```

Now the program can be run from anywhere.

Step:53

```
[fossilab@fossilab hello-2.7]$ hello
```

```
Hello, world!
```



14.3 Result/Observation

Thus the program for compiling from source is performed successfully.

14.4 Viva

1. What is the syntax of automake?
(A) automake [OPTION]....[Makefile]
(B) automake [OPTION]
(C) automake [Makefile]
(D) automake
2. What is the command that Forces the update of standard files?
(A) -a or -add-missing
(B) -libdir = DIRECTORY
(C) -c or -copy
(D) -f or -force-missing
3. What is the command to build a binary distribution?
(A) cpack -config CPackConfig.cmake
(B) set (CTEST_PROJECT_NAME "Tutorial")
(C) cpack -config CPackSourceConfig.cmake
(D) None of the above
4. Which one is the properties of the Ant?
(A) Basedir (B) Baiddir (C) Bootdir (D) None
5. List some basic functions performed by Ant.
(A) Compiling Java code into bytecode
(B) Placing this bytecode in a package
(C) Deployment to production systems
(D) All of these

15 Kernel Configuration, Compilation and Installation

15.1 Aim

Download / access the latest kernel source code from kernel.org, compile the kernel and install it in the local system. Try to view the source code of the kernel.

15.2 Overview

15.2.1 Algorithm:

Follow the following Steps:

Step 1: Open the browser and type the following command:

HTTP://192.168.105.254/CS2406/Software requirements for Lab exercises/ ->

select the kernel source code (ex no 1)->

display the linux-2.6.35.7.tar.gz->

select linux-2.6.35.7.tar.gz link->

save the folder in Downloads directory->

go to Download directory from places in menubar->

extract linux-2.6.35.7.tar.gz folder on that same directory.

Step 2: Display the user name

```
[fossilab@fossilab ~]$ uname -r
```

```
2.6.35.6-45.fc14.i686
```

Step 3: Enter the Download Directory and Display password of the Directory and display the list

```
[fossilab@fossilab ~]$ cd Downloads/
```

```
[fossilab@fossilab Downloads]$ pwd
```

```
/home/fossilab/Downloads
```

```
[fossilab@fossilab Downloads]$ ls -l
```

```
total 93332
```

```
-rw-rw-r-- 1 fossilab fossilab 344011 Jun 28 01:50 04524284.pdf
```

```
-rw-rw-r-- 1 fossilab fossilab 251225 Jun 28 01:36
```

```
91-US-31-1.Cloud.Computing.pdf
```

```
-rw-rw-r-- 1 fossilab fossilab 556032 Jun 28 01:41 ABSTRACT and pro_vidhya.doc
```

```
-rw-rw-r-- 1 fossilab fossilab 429466 Jun 28 01:37 computing-whitepaper.pdf
```

```
-rw-rw-r-- 1 fossilab fossilab 99019 Aug 1 01:03 Criterion-8.docx
```

```
-rw-rw-r-- 1 fossilab fossilab 110194 Aug 1 01:03 Criterion 9(2).docx
```

```
-rw-rw-r-- 1 fossilab fossilab 110194 Aug 1 01:03 Criterion 9.docx
```

```
-rw-rw-r-- 1 fossilab fossilab 512000 Aug 1 01:03 criter XP12-chandran.doc
```

```
-rw-rw-r-- 1 fossilab fossilab 422400 Aug 1 01:03 criter XP12_Mech.doc
```

```
-rwxrwxr-x 1 fossilab fossilab 493564 Jun 7 2011 Downloads.exe
```

```
-rw-rw-r-- 1 fossilab fossilab 634100 Jun 28 00:13 EJSR_64_2_05.pdf
```

```
-rw-rw-r-- 1 fossilab fossilab 237418 Jun 28 00:13 EJSR_64_2_14.pdf
```

```

-rw-rw-r- 1 fosslab fosslab 54227 Jun 28 01:39 EJSR_74_3_04.pdf
-rw-rw-r- 1 fosslab fosslab 527523 Jun 28 01:44 EJSR_77_1_06.pdf
-rw-rw-r- 1 fosslab fosslab 147175 Jul 9 01:55 foss-lab-manual-p1-1.0-rc1.pdf
-rw-rw-r- 1 fosslab fosslab 20228 Jun 28 01:50 Gartner Data Mining Addtl.pdf
-rw-rw-r- 1 fosslab fosslab 12253 Aug 11 14:02 HP-LaserJet-laserjet.ppd
-rw-rw-r- 1 fosslab fosslab 219237 Jun 28 00:10 kdd98_elder_abbott_nopics_bw.pdf
-rw-rw-r- 1 fosslab fosslab 88323744 Aug 16 14:27 linux-2.6.35.7.tar.gz
-rw-rw-r- 1 fosslab fosslab 60416 Jul 14 04:56 newFOC LP.doc
-rw-rw-r- 1 fosslab fosslab 43520 Jun 28 01:40 ProjectTitles.doc
-rw-rw-r- 1 fosslab fosslab 180964 Jun 28 01:57 sensor-route-security.pdf
-rw-rw-r- 1 fosslab fosslab 56320 Jul 18 01:55 ssLABabet outcome.doc
-rw-rw-r- 1 fosslab fosslab 142336 Jul 18 01:53 ss Lesson-plan.doc
-rw-rw-r- 1 fosslab fosslab 14540 Jul 30 04:55 st_newmark3(2).jsp
-rw-rw-r- 1 fosslab fosslab 14702 Jul 30 04:45 st_newmark3.jsp
-rw-rw-r- 1 fosslab fosslab 1517376 Jul 28 02:36 wrar420.exe

```

Step 4: Enter the linux-2.6.35.7 Directory and display the password of the linux-2.6.35.7 and display the list of that linux-2.6.35.7 directory

```
[fosslab@fosslab Downloads]$ cd linux-2.6.35.7/
```

```
[fosslab@fosslab linux-2.6.35.7]$ pwd
```

```
/home/fosslab/Downloads/linux-2.6.35.7
```

```
[fosslab@fosslab linux-2.6.35.7]$ ls -l
```

```
total 456
```

```

drwxrwxr-x 25 fosslab fosslab 4096 Sep 28 2010 arch
drwxrwxr-x 2 fosslab fosslab 4096 Sep 28 2010 block
-rw-rw-r- 1 fosslab fosslab 18693 Sep 28 2010 COPYING
-rw-rw-r- 1 fosslab fosslab 94031 Sep 28 2010 CREDITS
drwxrwxr-x 3 fosslab fosslab 4096 Sep 28 2010 crypto
drwxrwxr-x 85 fosslab fosslab 12288 Sep 28 2010 Documentation
drwxrwxr-x 89 fosslab fosslab 4096 Sep 28 2010 drivers
drwxrwxr-x 36 fosslab fosslab 4096 Sep 28 2010 firmware
drwxrwxr-x 72 fosslab fosslab 4096 Sep 28 2010 fs
drwxrwxr-x 20 fosslab fosslab 4096 Sep 28 2010 include
drwxrwxr-x 2 fosslab fosslab 4096 Sep 28 2010 init
drwxrwxr-x 2 fosslab fosslab 4096 Sep 28 2010 ipc
-rw-rw-r- 1 fosslab fosslab 2440 Sep 28 2010 Kbuild
drwxrwxr-x 8 fosslab fosslab 4096 Sep 28 2010 kernel
drwxrwxr-x 6 fosslab fosslab 4096 Sep 28 2010 lib
-rw-rw-r- 1 fosslab fosslab 174535 Sep 28 2010 MAINTAINERS
-rw-rw-r- 1 fosslab fosslab 51251 Sep 28 2010 Makefile
drwxrwxr-x 2 fosslab fosslab 4096 Sep 28 2010 mm
drwxrwxr-x 50 fosslab fosslab 4096 Sep 28 2010 net
-rw-rw-r- 1 fosslab fosslab 17459 Sep 28 2010 README
-rw-rw-r- 1 fosslab fosslab 3371 Sep 28 2010 REPORTING-BUGS

```



```
drwxrwxr-x 7 fosslab fosslab 4096 Sep 28 2010 samples
drwxrwxr-x 12 fosslab fosslab 4096 Sep 28 2010 scripts
drwxrwxr-x 7 fosslab fosslab 4096 Sep 28 2010 security
drwxrwxr-x 21 fosslab fosslab 4096 Sep 28 2010 sound
drwxrwxr-x 4 fosslab fosslab 4096 Sep 28 2010 tools
drwxrwxr-x 2 fosslab fosslab 4096 Sep 28 2010 usr
drwxrwxr-x 3 fosslab fosslab 4096 Sep 28 2010 virt
```

Step 5:Open the Makefile in vi editor and type the extraversion directory

```
[fosslab@fosslab linux-2.6.35.7]$ vi Makefile
```

```
VERSION = 2
PATCHLEVEL = 6
SUBLEVEL = 35
EXTRAVERSION = .7-veni
NAME = Yokohama
```

Step 6:To make the menuconfiguration

```
[fosslab@fosslab linux-2.6.35.7]$ make menuconfig
```

```
HOSTCC scripts/basic/fixdep
HOSTCC scripts/basic/docproc
HOSTCC scripts/basic/hash
HOSTCC scripts/kconfig/conf.o
HOSTCC scripts/kconfig/kxgettext.o
HOSTCC scripts/kconfig/lxdialog/checklist.o
HOSTCC scripts/kconfig/lxdialog/inputbox.o
HOSTCC scripts/kconfig/lxdialog/menubox.o
HOSTCC scripts/kconfig/lxdialog/textbox.o
HOSTCC scripts/kconfig/lxdialog/util.o
HOSTCC scripts/kconfig/lxdialog/yesno.o
HOSTCC scripts/kconfig/mconf.o
SHIPPED scripts/kconfig/zconf.tab.c
SHIPPED scripts/kconfig/lex.zconf.c
SHIPPED scripts/kconfig/zconf.hash.c
HOSTCC scripts/kconfig/zconf.tab.o
HOSTLD scripts/kconfig/mconf
scripts/kconfig/mconf arch/x86/Kconfig
#
# using defaults found in /boot/config-2.6.35.6-45.fc14.i686
#
#
# configuration written to .config
#
*** End of Linux kernel configuration.
*** Execute 'make' to build the kernel or try 'make help'.
```

Step 7:Display the General setup screen and do the following steps

file systems->DOS/FAT/NT FILE SYSTEM->NTFS FILE SYSTEMS press space
bar->NTFS FILE SUPPORT press the space bar->exit

Step 8: To make the bootImage from the x86 boot directory

```
[fossilab@fossilab linux-2.6.35.7]$make bzImage
```

Step 9: Login in to the super user(root)

```
[fossilab@fossilab linux-2.6.35.7]$ su
```

Password: (admin123)

Step 10: To copy the bootlinuzImage from the x86 boot directory in to the vmlinuz-2.6.35.7

directory

```
[root@fossilab linux-2.6.35.7]# cp arch/x86/boot/bzImage /boot/vmlinuz-2.6.35.7-veni
```

Step 11: Exit from the super user

```
[root@fossilab linux-2.6.35.7]# exit
```

exit

Step 12: To make the modules in linux-2.6.35.7 and Display list of modules are available in modules

```
[fossilab@fossilab linux-2.6.35.7]$ make modules
```

```
[fossilab@fossilab linux-2.6.35.7]$ ls -l /lib/modules
```

total 8

```
drwxr-xr-x. 6 root root 4096 May 29 02:54 2.6.35.6-45.fc14.i686
```

```
drwxr-xr-x. 6 root root 4096 May 29 02:48
```

```
2.6.35.6-45.fc14.i686.PAE
```

Step 13: Login in to the super user(root)

```
[fossilab@fossilab linux-2.6.35.7]$ su
```

Password:(admin123)

Step 14: Modules are install into the linux-2.6.35.7 on the super user and Display the available directories on the linux-2.6.35.7

```
[root@fossilab linux-2.6.35.7]# make modules_install
```

```
[root@fossilab linux-2.6.35.7]# ls -l /lib/modules
```

total 12

```
drwxr-xr-x. 6 root root 4096 May 29 02:54 2.6.35.6-45.fc14.i686
```

```
drwxr-xr-x. 6 root root 4096 May 29 02:48
```

```
2.6.35.6-45.fc14.i686.PAE
```

```
drwxr-xr-x 3 root root 4096 Aug 17 11:49 2.6.35.7-veni
```

Step 15: stored all the hardwares and directories in to the initram directory

```
/////////[root@fossilab linux-2.6.35.7]# mkinitrd
```

```
/boot/initramfs-2.6.35.7-veni.img 2.6.35.7-veni
```

Step 16: Enter in to the boot directory and Display the list of files are available on the boot directoy

```
[root@fossilab linux-2.6.35.7]# cd /boot
```

```
[root@fossilab boot] ls -l
```

total 61848

```

-rw-r--r-. 1 root root 114968 Oct 18 2010
config-2.6.35.6-45.fc14.i686
-rw-r--r-. 1 root root 115205 Oct 18 2010
config-2.6.35.6-45.fc14.i686.PAE
drwxr-xr-x. 3 root root 4096 May 29 01:06 efi
drwxr-xr-x. 2 root root 4096 May 29 00:53 extlinux
drwxr-xr-x. 2 root root 4096 May 29 02:56 grub
-rw-r--r-. 1 root root 13507699 May 29 02:46
initramfs-2.6.35.6-45.fc14.i686.img
-rw-r--r-. 1 root root 13502690 May 29 02:47
initramfs-2.6.35.6-45.fc14.i686.PAE.img
-rw-r--r- 1 root root 11005350 Aug 17 11:51
initramfs-2.6.35.7-veni.img
-rw-r--r-. 1 root root 1106328 May 29 01:52 initrd-plymouth.img
-rw-r--r-. 1 root root 1681526 Oct 18 2010
System.map-2.6.35.6-45.fc14.i686
-rw-r--r-. 1 root root 1709576 Oct 18 2010
System.map-2.6.35.6-45.fc14.i686.PAE
-rwxr-xr-x. 1 root root 3696448 Oct 18 2010
vmlinuz-2.6.35.6-45.fc14.i686
-rwxr-xr-x. 1 root root 3761568 Oct 18 2010
vmlinuz-2.6.35.6-45.fc14.i686.PAE
-rw-r--r- 1 root root 3677056 Aug 17 10:58 vmlinuz-2.6.35.7-veni
-rw-r--r-. 1 root root 571311 Oct 12 2010 xen-4.0.1.gz
lrwxrwxrwx. 1 root root 12 May 29 00:40 xen.gz -> xen-4.0.1.gz
-rw-r--r-. 1 root root 8840980 Oct 12 2010 xen-syms-4.0.1
Step 17: Enter in to the grub file on the boot directory
[root@fossilab boot]# cd /boot/grub
Step 18: open the grub.conf file from grub directory
[root@fossilab grub]# vi grub.conf
add end of the file line
title Fedora (2.6.35.7-veni)
root (hd0,0)
kernel /boot/vmlinuz-2.6.35.7-mykernel ro
root=UUID=6c37c0aa-4b4c-4bbe-a235-d9149be80d24
rd_NO_LUKS rd_NO_LVM rd_NO_MD rd_NO_DM LANG=en_US.UTF-8
SYSFONT=latacyrhebsun16
KEYTABLE=us rhgb quiet
initrd /boot/initramfs-2.6.35.7-veni.img
copy this statement form title fedora
ro root=UUID=6c37c0aa-4b4c-4bbe-a235-d9149be80d24 rd_NO_LUKS rd_NO_LVM
rd_NO_MD
rd_NO_DM LANG=en_US.UTF-8 SYSFONT=latacyrheb-sun16 KEYTABLE=us rhgb

```

quiet

Step 19: Reboot the system

```
[root@fossilab grub]# reboot
```

15.3 Result/Observation

Thus to download / access the latest kernel source code from kernel.org, compiling the kernel and install it in the local system and trying to view the source code of the kernel is done successfully.

15.4 Viva

1. The dmesg command
 - (A) Shows user login logoff attempts
 - (B) Shows the syslog file for info messages
 - (C) kernel log messages
 - (D) Shows the daemon log messages
2. Which command is used to display the unix version
 - (A) uname -r
 - (B) uname -n
 - (C) uname -t
 - (D) kernel
3. Which option of ls command used to view file inode number
 - (A) -l
 - (B) -o
 - (C) -a
 - (D) -i

16 GUI Programming

16.1 Aim

Create scientific calculator using GTK.

16.2 Overview

The prerequisites for this experiment are pyGTK. Install pyGTK by following instructions on this website - <http://www.pygtk.org/downloads.html>.

16.2.1 Python code

```
import gi
gi.requireversion('Gtk','3.0')
from gi.repository import Gtk
from math import *
class calcWindow(Gtk.Window) :
def init(self):
Gtk.Window.init(self,title="Scientific Calculator")
outerbox = Gtk.Box(spacing=10,orientation = Gtk.Orientation.VERTICAL)
self.add(outerbox)
entry = Gtk.Entry( )
outerbox.packstart(entry,True,True,0)
grid = Gtk.Grid( )
outerbox.packstart (grid,True,True,0)
defbuttonclicked(button):
text = entry.gettext( )
text = text+button.props.label
entry.settext(text)
defcleartext(self):
entry.settext("")
def evaluate(self) :
eq = entry.gettext()
entry.settext(str(eval(eq)))
def delsingl(self):
text = entry.gettext()
text=text[: -1]
entry.settext(text)
#buttons
button9 = Gtk.Button(label = "9" ) ;
button8 = Gtk.Button(label= "8" ) ;
button7 = Gtk.Button(label="7") ;
delete = Gtk.Button(label="DEL") ;
```

```

ac = Gtk.Button(label="AC") ;
button4 = Gtk.Button(label="4") ;
button5 = Gtk.Button(label="5") ;
button6 = Gtk.Button(label="6") ;
multiply = Gtk.Button(label="7") ;
divide = Gtk.Button(label="/" ) ;
button1 = Gtk.Button(label="1" ) ;
button2 = Gtk.Button(lael="2" ) ;
button3 = Gtk.Button(label="3" ) ;
plus = Gtk.Button(label="+" ) ;
minus = Gtk.Button(label="-" ) ;
button = Gtk.Button(label="9" ) ;
ans = Gtk.Button(label="=" ) ;
dot = Gtk.Button(label="." )
openbr = Gtk.Button(label="(" )
closebr = Gtk.Button(label=")" )
clear = Gtk.Button(label="C" )
root = Gtk.Button(label="root" )
log = Gtk.Button(label="log" )
other buttons =(divide , plus , minus , multiply )
for button name in other buttons
buttonname.connect('clicked',buttonclicked)
digits=button1 , button2 , button3 , button4 , button5 , button6 , button7 , button8
, button9 , button0 , openbr , closebr )
for i in digits :
i.connect('clicked',buttonclicked)
#connecting special buttons
ans.connect('clicked',evaluate )
#ac.connect('clicked',clear)
delete.connect('clicked',delsingle)
clear.connect("clicked",cleartext)
grid.add(button9)
grid.attach(button8 , 1 , 0 , 1 , 1 )
grid.attachnextto( button7 , button8 ,Gtk . PositionType .RIGHT, 1 , 1 )
grid.attachnextto ( divide , button7 ,Gtk . PositionType .RIGHT, 1 , 1 )
grid.attachnextto( button6 , button9 ,Gtk . PositionType .BOTTOM, 1 , 1 )
grid.attachnextto ( button5 , button6 ,Gtk . PositionType .RIGHT, 1 , 1 )
grid.attachnextto ( button4 , button5 ,Gtk . PositionType .RIGHT, 1 , 1 )
grid.attachnextto ( multiply , button4 ,Gtk . PositionType .RIGHT, 1 , 1 )
grid.attach nextto ( button3 , button6 ,Gtk . PositionType .BOTTOM, 1 , 1 )
grid.attachnextto ( button2 , button3 ,Gtk . PositionType .RIGHT, 1 , 1 )
grid.attachnextto ( button1 , button2 ,Gtk . PositionType .RIGHT, 1 , 1 )
grid.attachnextto (minus , button1 ,Gtk . PositionType .RIGHT, 1 , 1 )

```

```

grid.attachnextto ( button0 , button3 ,Gtk . PositionType .BOTTOM, 1 , 1 )
grid.attachnextto ( dot , button0 ,Gtk . PositionType .RIGHT, 1 , 1 )
grid.attachnextto ( ans , dot ,Gtk . PositionType .RIGHT, 1 , 1 )
grid.attachnextto ( plus , ans ,Gtk . PositionType .RIGHT, 1 , 1 )
grid.attachnextto(openbr,button9 ,Gtk . PositionType .TOP, 1 , 1 )
grid.attachnextto(closebr, button8 ,Gtk . PositionType .TOP, 1 , 1 )
grid.attachnextto(clear, button7 ,Gtk . PositionType .TOP, 1 , 1 )
grid.attachnextto delete, divide ,Gtk . PositionType .TOP, 1 , 1 )
calcWindow = calcWindow ( )
calcWindow . connect ( 'delete-event ' ,Gtk . main quit )
calcWindow . showall( )
Gtk .main( )

```

16.3 Sample input/output



16.4 Result/Observation

The python script for creating a scientific calculator was made and the script was run on Ubuntu 15.10 (Wiley Werewolf). Python 2.7.13 was used along with pyGTK+ 3.0.

16.5 Viva

1. Which is/are of the following is/are true about PHP-GTK ?
 - (A) PHP-GTK provides an object-oriented interface to GTK+ classes and functions
 - (B) PHP-GTK is a library for PHP programming language which implements language bindings.
 - (C) Only A
 - (D) Both [A] and [B]

2. What does `-5 % 11` evaluate to?
(A) +5 (B) -5 (C) +11 (D) -11
3. The function `divmod(a,b)`, where both 'a' and 'b' are integers is evaluated as:
(A) `(a%b, a//b)`
(B) `(a//b, a%b)`
(C) `(a//b, a*b)`
(D) `(a/b, a%b)`
4. How do you specify user data to a signal?
(A) `queue_draw_area(x_beg, y_beg, x_end, y_end)`
(B) `label = gtk.Label("MyLabel")`
(C) `widget.connect("my_signal", handler, userdata [, ...])`
(D) `w.connect("delete-event", on_win_delete)`

17 Installing Various Software Packages

17.1 Aim

Installing various software packages. Either the package is yet to be installed or an older version is present.

- Install samba and share files to windows
- Install Common Unix Printing System(CUPS)

17.2 Overview

17.2.1 Installation of Samba:

Requirement: Internet for downloading application. The installation of samba is simple.

Step 1: Login to the super user by typing,
su

Step 2: Then install the samba by typing the following command,
yum install samba (Not needed if it is already installed) Step 3: Make the Samba server to run by using the following command
service smb start

17.2.2 Testing Communication of Linux with windows:

Requirement: Two systems, one is booted with Fedora and other is booted with Windows. Before going to accessing files of different environment, it is required to turn off the Security mechanisms of Linux.

a) To access Windows from Linux:

Step1 : Find the IP Address of Windows system (sample: 192.168.1.26)

Step2 : In Fedora , in places, type Ctrl +L and in the address bar type smb ://192.168.1.26

Step3 : It will ask for the following details

Pwd required for sharing C\$ on 192.168.1.26

User Name:administrator

Domain : cse

Password:(skrcse2011)

For getting the Domain name use the following steps in Windows:

In command Prompt type :

C:\> hostname

cse

Step4 : the available files and folders will be visible in Fedora.

b) To access Linux from Windows

Step1 : In fedora , check the status of samba using the following command

```
[fossilab@fossilab ~]$ su -
```

```
[fossilab@fossilab ~]$ service smb status
```

```
[fossilab@fossilab ~]$ rpm -qa --grep samba
```

```
[fossilab@fossilab ~]$ service smb restart (if not running already)
```

Step2: Now go to Windows Systems, Type the IP address of Fedora System in Run prompt as Run ://192.168.1.245

Step 3: To Map linux user into Samba user do the following

```
[fossilab@fossilab ~]$ useradd winuser
```

```
[fossilab@fossilab ~]$ passwd winusersmbpasswd -a winuser
```

Asks for password, Now type admin123 or any

Step4: To extend the functionality of Samba

```
[fossilab@fossilab ~]$ vi /etc/samba/smb.conf
```

Step5: Make the directory skrmicrosoft with machine names

```
[fossilab@fossilab ~]$ mkdir /software/skrmicrosoft/ Adobe, IBM, Oracle -p
```

Step6: To configure the samba file

```
[fossilab@fossilab ~]$ vi smb.conf
```

Type the following in last line

```
[software]
```

```
Comment=Common Software Repository
```

```
Path=/software
```

```
Writable=no
```

```
Readonly=yes
```

```
Public=yes
```

```
Browsable=yes
```

Save the file

Step7: To restart the samba service

```
[fossilab@fossilab ~]$ service smb restart
```

Now check in windows the folder software will be visible with subfolders Adobe, IBM, Oracle

17.2.3 Installation of CUPS

Step1: Open the terminal and log in.

Step2: Type the following command

```
$ rpm -qa | grep cups.
```

Step3: Type the command

```
$ rpm -qi cups
```

Step4: Then using clear command clear the screen.

Step5: Login to super user .

```
$su
```

Step6: Check the status of the cups using the following command
service cups status

Step7: Start the service of cups using the following command
 service cups status
 Step8: Check the status of the cups again.
 Step9: Click Mozilla browser and type
 http://localhost:631/
 Step10: Select the option Adding printer and classes.
 Step11: Select the option “Add printer”
 Step12: Give the root username and root password.
 Step13: Select the printer type and select continue option.
 Step14: Give the printer name, location and description and give continue option.
 Step15: Select the maker of printer and select the continue option.
 Step16: Select the model of printer and select the option -Add printer
 Step17: Select the paper size and set the default options.
 Step18: Click on the printer name and see the jobs that are pending.

17.3 Result/Observation

The program to share the file using SAMBA and the program Installing Common UNIX Printing System (CUPS) and testing is successfully Completed.

17.4 Viva

1. Which of the following commands tests the syntax of the default Samba configuration file?
 Please select the best answer.
 (A) smbclient -L //localhost
 (B) testparm //localhost
 (C) smbclient /etc/smb.conf
 (D) testparm /etc/smb.conf
2. Examples of system programs includes
 (A)operating system of computer
 (B)trace program
 (C)compiler
 (D)all of above
3. Set of programs which consist of full set of documentations is termed as
 (A)database packages (B)file packages (C)bus packages (D)software packages