LINUX PROGRAMMING

LABORATORY MANUAL

B.TECH
(III YEAR – I SEM)
(2019-20)

DEPARTMENT OF INFORMATION TECHNOLOGY

MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY

(Autonomous Institution – UGC, Govt. of India)

Recognized under 2(f) and 12 (B) of UGC ACT 1956

Affiliated to JNTUH, Hyderabad, Approved by AICTE - Accredited by NBA & NAAC – 'A' Grade - ISO 9001:2015 Certified)

Maisammaguda, Dhulapally (Post Via. Hakimpet), Secunderabad – 500100, Telangana State, India

DEPARTMENT OF INFORMATION TECHNOLOGY

VISION

- To improve the quality of technical education that provides efficient software engineers with an attitude to adapt challenging IT needs of local, national and international arena, through teaching and interaction with alumni and industry.

MISSION

- Department intends to meet the contemporary challenges in the field of IT and is playing a vital role in shaping the education of the 21st century by providing unique educational and research opportunities.

PROGRAMME EDUCATIONAL OBJECTIVES (PEOs)

PEO1 – ANALYTICAL SKILLS

To facilitate the graduates with the ability to visualize, gather information, articulate, analyze, solve complex problems, and make decisions. These are essential to address the challenges of complex and computation intensive problems increasing their productivity.

PEO2 – TECHNICAL SKILLS

To facilitate the graduates with the technical skills that prepare them for immediate employment and pursue certification providing a deeper understanding of the technology in advanced areas of computer science and related fields, thus encouraging to pursue higher education and research based on their interest.

PEO3 – SOFT SKILLS

To facilitate the graduates with the soft skills that include fulfilling the mission, setting goals, showing self-confidence by communicating effectively, having a positive attitude, get involved in team-work, being a leader, managing their career and their life.

PEO4 – PROFESSIONAL ETHICS

To facilitate the graduates with the knowledge of professional and ethical responsibilities by paying attention to grooming, being conservative with style, following dress codes, safety codes, and adapting technological advancements.

PROGRAM SPECIFIC OUTCOMES (PSOs)

After the completion of the course, B. Tech Information Technology, the graduates will have the following Program Specific Outcomes:

1. **Fundamentals and critical knowledge of the Computer System:-** Able to Understand the working principles of the computer System and its components , Apply the knowledge to build, asses, and analyze the software and hardware aspects of it .

2. **The comprehensive and Applicative knowledge of Software Development:** Comprehensive skills of Programming Languages, Software process models, methodologies, and able to plan, develop, test, analyze, and manage the software and hardware intensive systems in heterogeneous platforms individually or working in teams.

3. **Applications of Computing Domain & Research:** Able to use the professional, managerial, interdisciplinary skill set, and domain specific tools in development processes, identify the research gaps, and provide innovative solutions to them.

PROGRAMOUTCOMES (POs)

Engineering Graduates should possess the following:

1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **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.
3. **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.
4. **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.
5. **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.
6. **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.
7. **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.
8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **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.
11. **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 multi disciplinary environments.
12. **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.

DEPARTMENT OF INFORMATION TECHNOLOGY

GENERAL LABORATORY INSTRUCTIONS

1. Students are advised to come to the laboratory at least 5 minutes before (to the starting time), those who come after 5 minutes will not be allowed into the lab.
2. Plan your task properly much before to the commencement, come prepared to the lab with the synopsis / program / experiment details.
3. Student should enter into the laboratory with:

a. Laboratory observation notes with all the details (Problem statement, Aim, Algorithm, Procedure, Program, Expected Output, etc.,) filled in for the lab session.

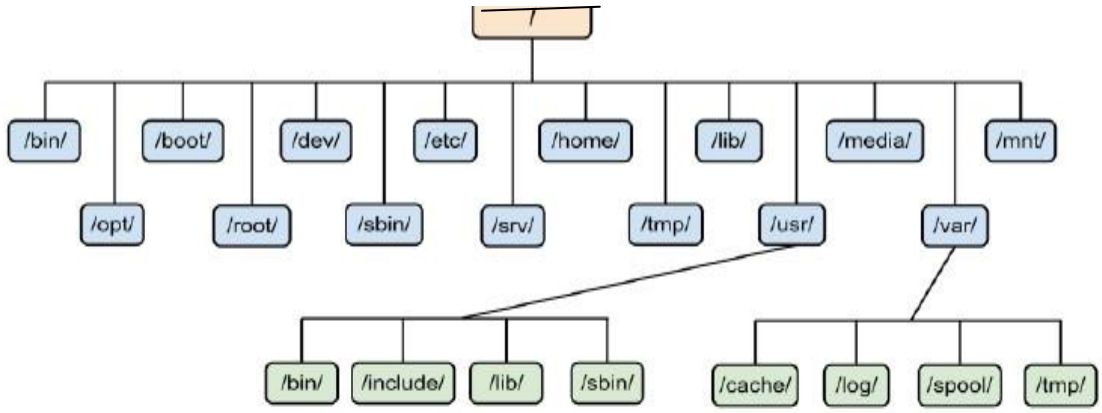
b. Laboratory Record updated up to the last session experiments and other utensils (if any) needed in the lab.

c. Proper Dress code and Identity card.

4. Sign in the laboratory login register, write the TIME-IN, and occupy the computer system allotted to you by the faculty.
5. Execute your task in the laboratory, and record the results / output in the lab observation note book, and get certified by the concerned faculty.
6. All the students should be polite and cooperative with the laboratory staff, must maintain the discipline and decency in the laboratory.
7. Computer labs are established with sophisticated and high end branded systems, which should be utilized properly.
8. Students / Faculty must keep their mobile phones in SWITCHED OFF mode during the lab sessions. Misuse of the equipment, misbehaviors with the staff and systems etc., will attract severe punishment.
9. Students must take the permission of the faculty in case of any urgency to go out; if anybody found loitering outside the lab / class without permission during working hours will be treated seriously and punished appropriately.
10. Students should LOG OFF/ SHUT DOWN the computer system before he/she leaves the lab after completing the task (experiment) in all aspects. He/she must ensure the system / seat is kept properly.

HEAD OF THE DEPARTMENT **PRINCIPAL**

INDEX



System Administration Tools

1. UNIX comes with its own tools such as SAM on HP-UX.
2. Suse Linux comes with Yast
3. Redhat Linux comes with its own gui tools called redhat-config-*.

However, editing text config file and typing commands are most popular options for sys admin work under UNIX and Linux.

UNIX Operating System Names

A few popular names:

1. HP-UX
2. IBM AIX
3. Sun Solairs
4. Mac OS X
5. IRIX

Linux Distribution Names

A few popular names:

1. Redhat Enterprise Linux
2. Fedora Linux
3. Debian Linux
4. Suse Enterprise Linux
5. Ubuntu Linux

To Install Ubuntu Linux – Complete Step by Step

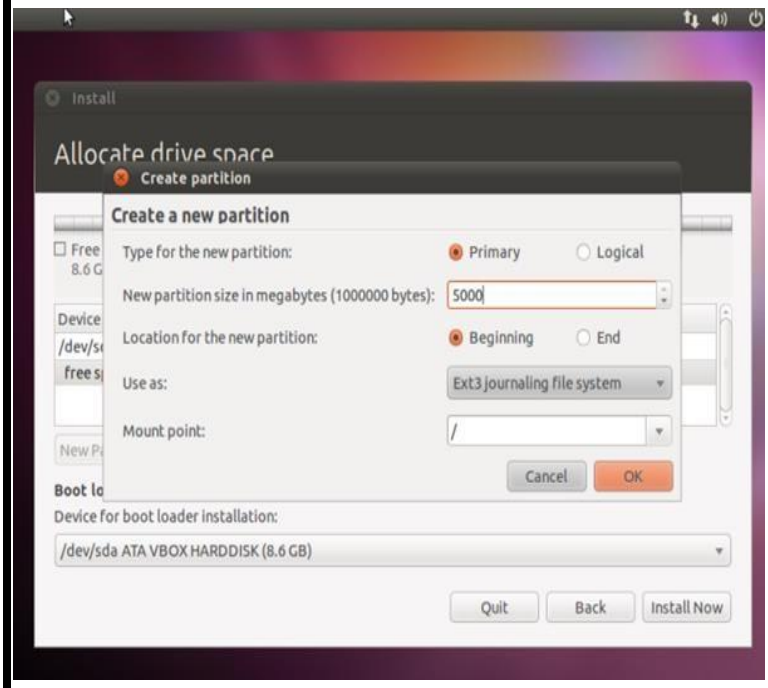
Step 1 : Insert the ubuntu cd in the cd drive and boot the computer from cd. First of all you will be prompted to select language. elect English or other language according to your preferences.

Step 2 : Now you will see ubuntu menu, you can choose **Try ubuntu without installing** option to try ubuntu without actually installing it on your hard drive. For installing ubuntu choose the second option **Install Ubuntu**.

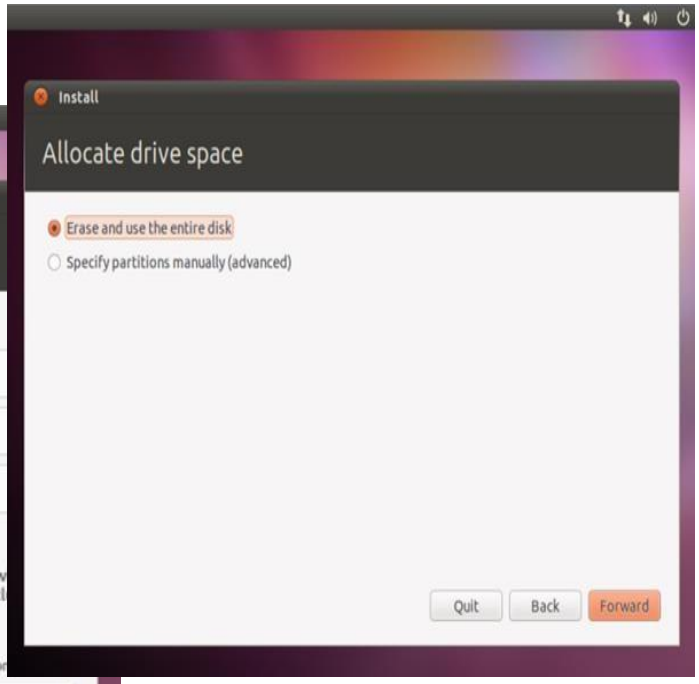
Step 3 : Ubuntu will start now initialize and after few minutes you can see the installation wizard.

Step 4 : Click Forward and it will check the **minimum requirements for running ubuntu** on your PC. If everything is fine you can see green colored tick marks. You can also select to download updates while installing and install some third party software. After selecting the things you want click forward.

Step 5 : Now you can choose either erase and use entire disk option or s**pecify partitions**

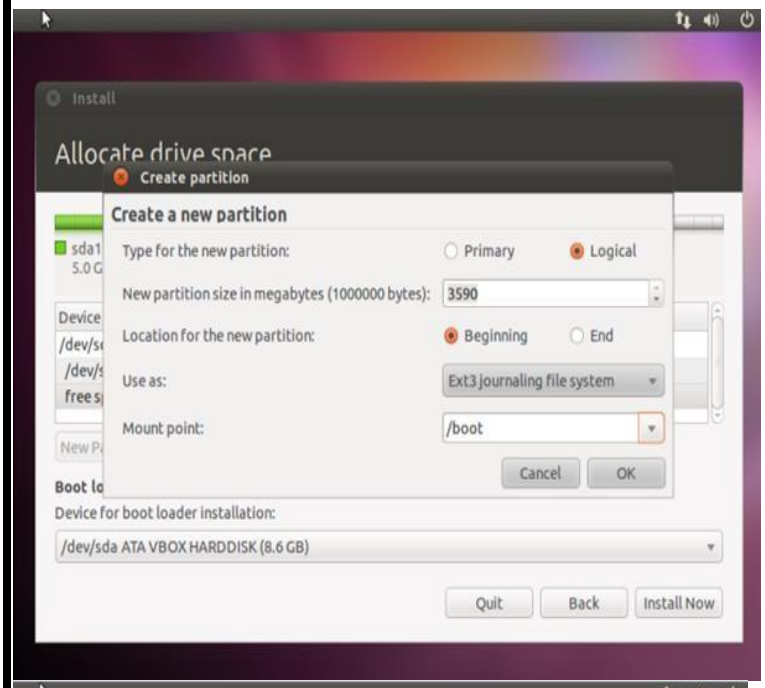


manually option. You can choose the 1st option if you just want linux to exist in your system. Else select second option. Now it will display the free space available for your pc.

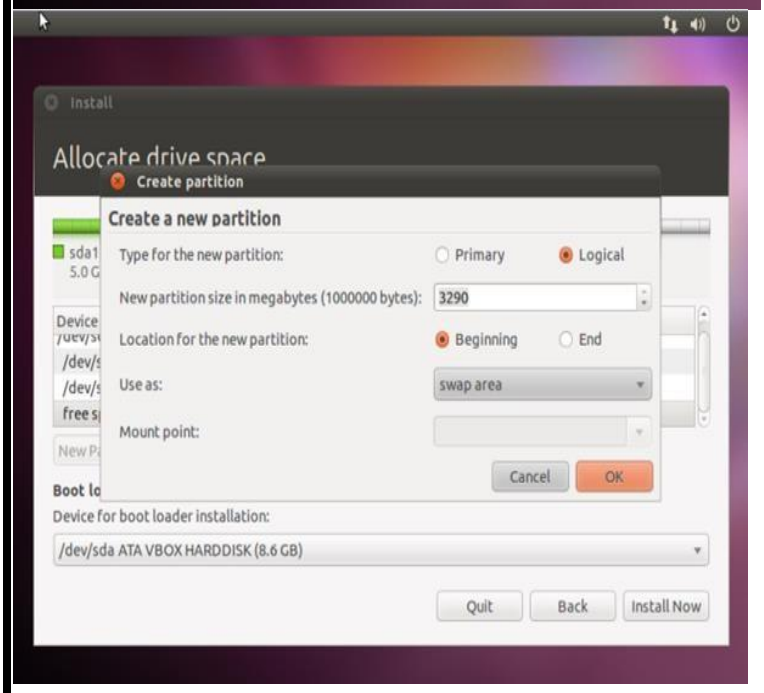


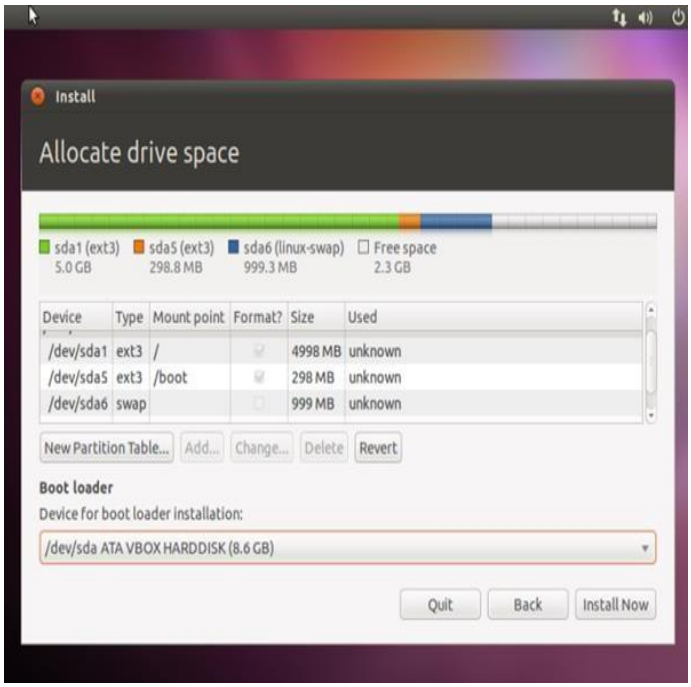
Select free space and click on Add option to create a new partition and choose partition type as primary, size around 70% of the free space available or choose anything like 10,000 or 20,000mb, use as ext3 journaling file system and select mount point as /.

Now again select free space from the table and click add option. Now select size to be around 300mb, use as ext3 journaling file system and select mount point as /boot.

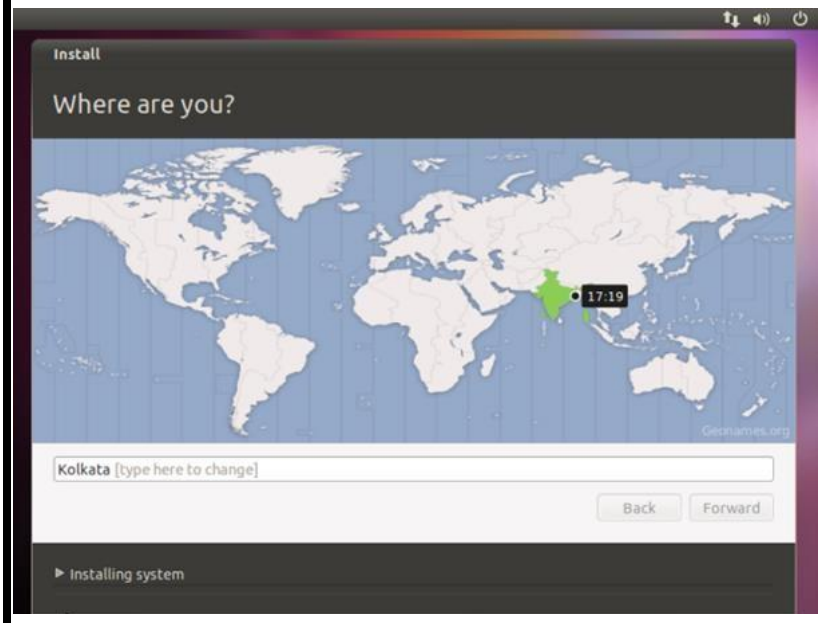


Now again select free space from the table and click add option. Now select size to be around twice the size of your ram that is around 1000 mb if your ram size is 512mb and select use as swap area and click ok.

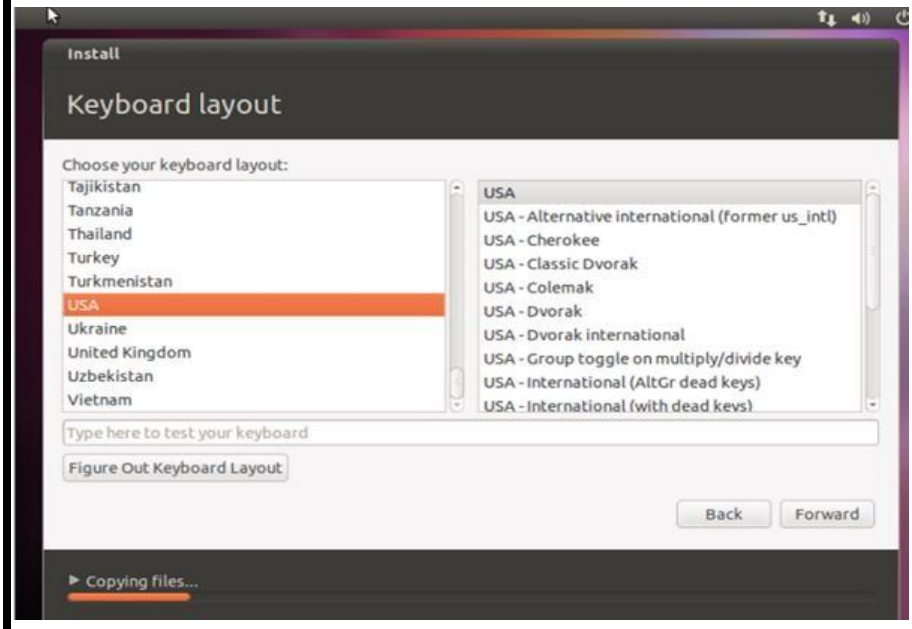




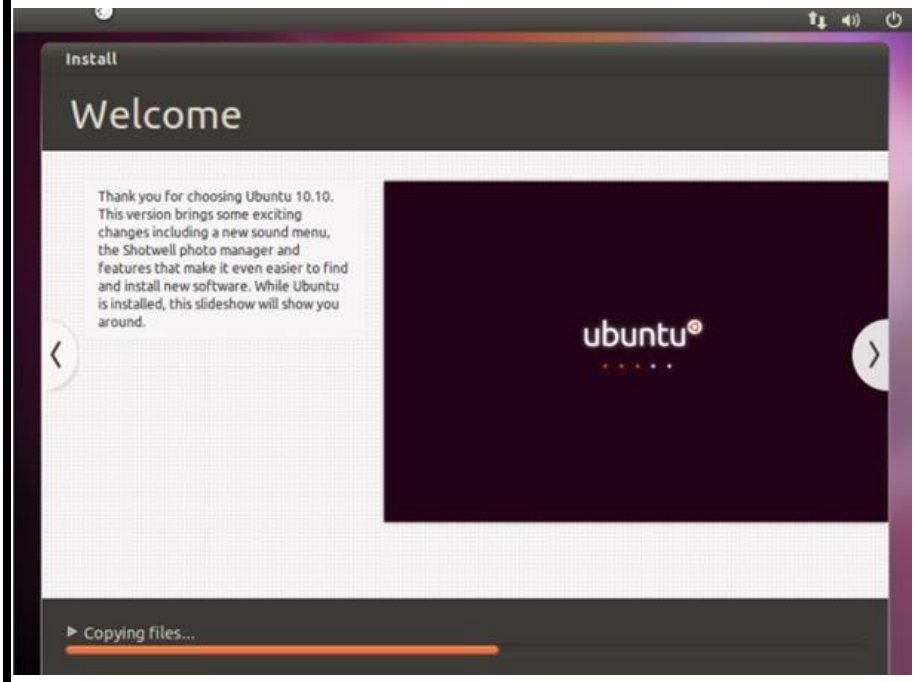
Step 6 : Click Install now button and then the wizard will ask you location. Select your location and click forward.



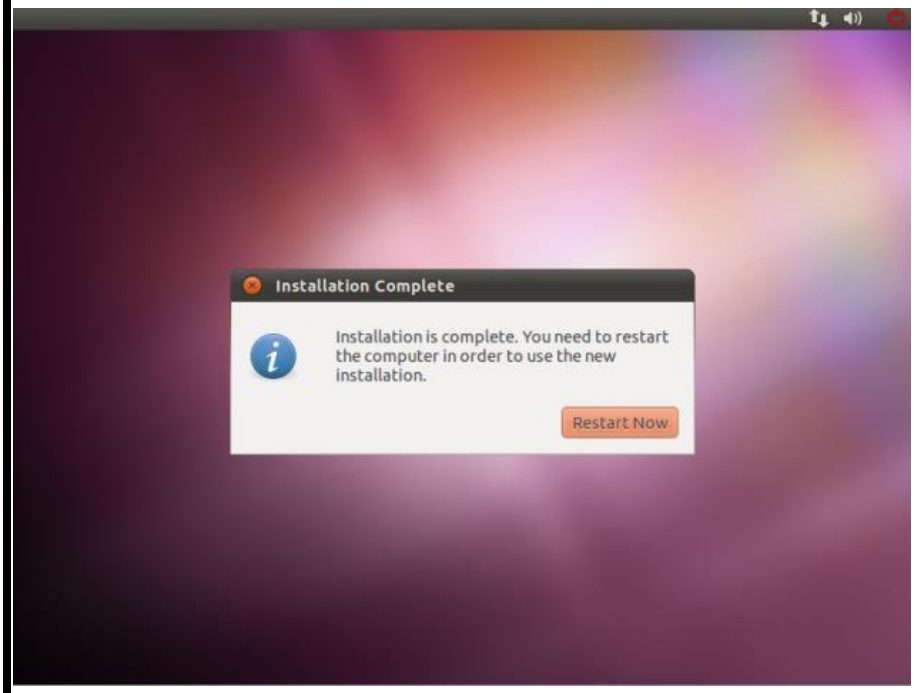
Step 7 : While you are selecting these options wizard will continue to copy files. Now select your desired keyboard layout and click forward.



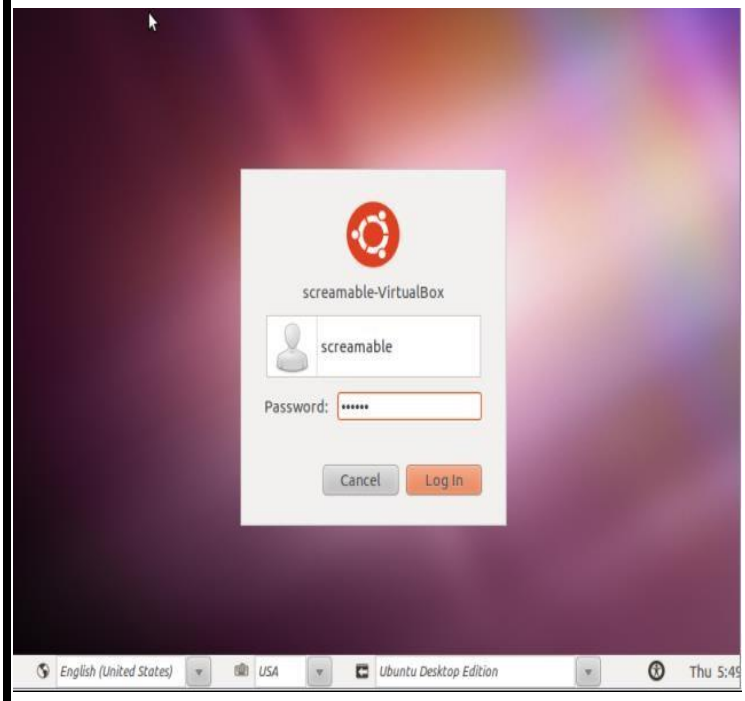
Step 8 : Now fill in the details about yourself. Fill your name, computer name, choose a username and create a password and click forward and let ubuntu copy all the essential files.

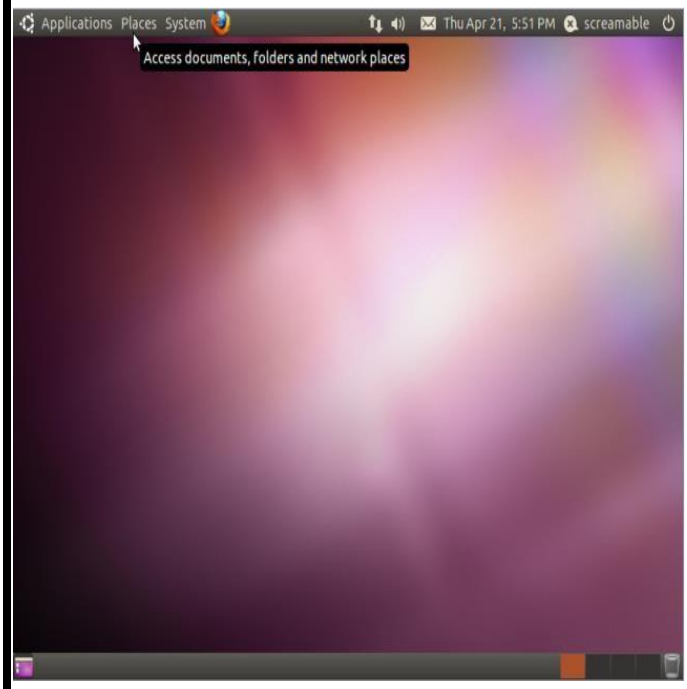


Step 9 : After all files have been copied and installed ubuntu will display a message saying that installation complete and click on restart button to restart your computer. Remove the cd from the cd drive.



Step 10 : After restarting your pc wait for the ubuntu to load and then it will display the login screen. Choose the user and enter password and click login.


screamable-VirtualBox
screamable
Password:
Cancel
Log In
English (United States)
USA
Ubuntu Desktop Edition
Thu 5:49


Applications Places System
Thu Apr 21, 5:51 PM
screamable
Access documents, folders and network places

A-Z Index of the Bash command line for Linux

a

alias Create an alias •
apropos Search Help manual pages (man -k)
apt-get Search for and install software packages (Debian/Ubuntu)
aptitude Search for and install software packages (Debian/Ubuntu)
aspell Spell Checker
awk Find and Replace text, database sort/validate/index

b

basename Strip directory and suffix from filenames
bash GNU Bourne-Again SHell
bc Arbitrary precision calculator language
bg Send to background
bind Set or display readline key and function bindings •
break Exit from a loop •
builtin Run a shell builtin
bzip2 Compress or decompress named file(s)

c

cal Display a calendar
case Conditionally perform a command
cat Concatenate and print (display) the content of files
cd Change Directory
cfdisk Partition table manipulator for Linux
chattr Change file attributes on a Linux file system
chgrp Change group ownership
chmod Change access permissions
chown Change file owner and group
chroot Run a command with a different root directory
chkconfig System services (runlevel)
cksum Print CRC checksum and byte counts
clear Clear terminal screen
cmp Compare two files
comm Compare two sorted files line by line
command Run a command - ignoring shell functions •
continue Resume the next iteration of a loop •
cp Copy one or more files to another location
cron Daemon to execute scheduled commands
crontab Schedule a command to run at a later time
csplit Split a file into context-determined pieces
curl Transfer data from or to a server
cut Divide a file into several parts

d

date Display or change the date & time
dc Desk Calculator
dd Convert and copy a file, write disk headers, boot records

ddrescue Data recovery tool
declare Declare variables and give them attributes •
df Display free disk space
diff Display the differences between two files
diff3 Show differences among three files
dig DNS lookup
dir Briefly list directory contents
dircolors Colour setup for `ls'
dirname Convert a full pathname to just a path
dirs Display list of remembered directories
dmesg Print kernel & driver messages
du Estimate file space usage

e

echo Display message on screen •
egrep Search file(s) for lines that match an extended expression
eject Eject removable media
enable Enable and disable builtin shell commands •
env Environment variables
eval Evaluate several commands/arguments
exec Execute a command
exit Exit the shell
expect Automate arbitrary applications accessed over a terminal
expand Convert tabs to spaces
export Set an environment variable
expr Evaluate expressions

f

false Do nothing, unsuccessfully
fdformat Low-level format a floppy disk
fdisk Partition table manipulator for Linux
fg Send job to foreground
fgrep Search file(s) for lines that match a fixed string
file Determine file type
find Search for files that meet a desired criteria
fmt Reformat paragraph text
fold Wrap text to fit a specified width.
for Expand *words*, and execute *commands*
format Format disks or tapes
free Display memory usage
fsck File system consistency check and repair
ftp File Transfer Protocol
function Define Function Macros
fuser Identify/kill the process that is accessing a file

g

gawk Find and Replace text within file(s)
getopts Parse positional parameters
grep Search file(s) for lines that match a given pattern
groupadd Add a user security group
groupdel Delete a group

```
  groupmod Modify a group
  groups   Print group names a user is in
  gzip     Compress or decompress named file(s)
h
  hash     Remember the full pathname of a name argument
  head     Output the first part of file(s)
  help     Display help for a built-in command •
  history  Command History
  hostname Print or set system name

i
  iconv    Convert the character set of a file
  id       Print user and group id's
  if       Conditionally perform a command
  ifconfig Configure a network interface
  install  Copy files and set attributes
  ip       Routing, devices and tunnels
j
  jobs     List active jobs •
  join     Join lines on a common field
k
  kill     Kill a process by specifying its PID
  killall  Kill processes by name
l
  let      Perform arithmetic on shell variables •
  link     Create a link to a file
  ln       Create a symbolic link to a file
  local    Create a function variable •
  locate   Find files
  logname  Print current login name
  logout   Exit a login shell •
  look     Display lines beginning with a given string
  lpc      Line printer control program
  lpr      Off line print
  lprint   Print a file
  lprintd  Abort a print job
  lprintq  List the print queue
  ls       List information about file(s)
  lsof     List open files
m
  make     Recompile a group of programs
  man      Help manual
  mkdir    Create new folder(s)
  mkfifo   Make FIFOs (named pipes)
  mkfile   Make a file
  mktemp   Make a temporary file
  more     Display output one screen at a time
  most     Browse or page through a text file
```

mount Mount a file system
mtools Manipulate MS-DOS files
mtr Network diagnostics (traceroute/ping)
mv Move or rename files or directories
mmv Mass Move and rename (files)

n

nc Netcat, read and write data across networks
netstat Networking connections/stats
nice Set the priority of a command or job
nl Number lines and write files
nohup Run a command immune to hangups
notify-send Send desktop notifications
nslookup Query Internet name servers interactively

o

open Open a file in its default application
op Operator access

p

passwd Modify a user password
paste Merge lines of files
ping Test a network connection
pgrep List processes by name
pkill Kill processes by name
popd Restore the previous value of the current directory
pr Prepare files for printing
printcap Printer capability database
printenv Print environment variables
printf Format and print data •
ps Process status
pushd Save and then change the current directory
pv Monitor the progress of data through a pipe
pwd Print Working Directory

q

quota Display disk usage and limits
quotacheck Scan a file system for disk usage

r

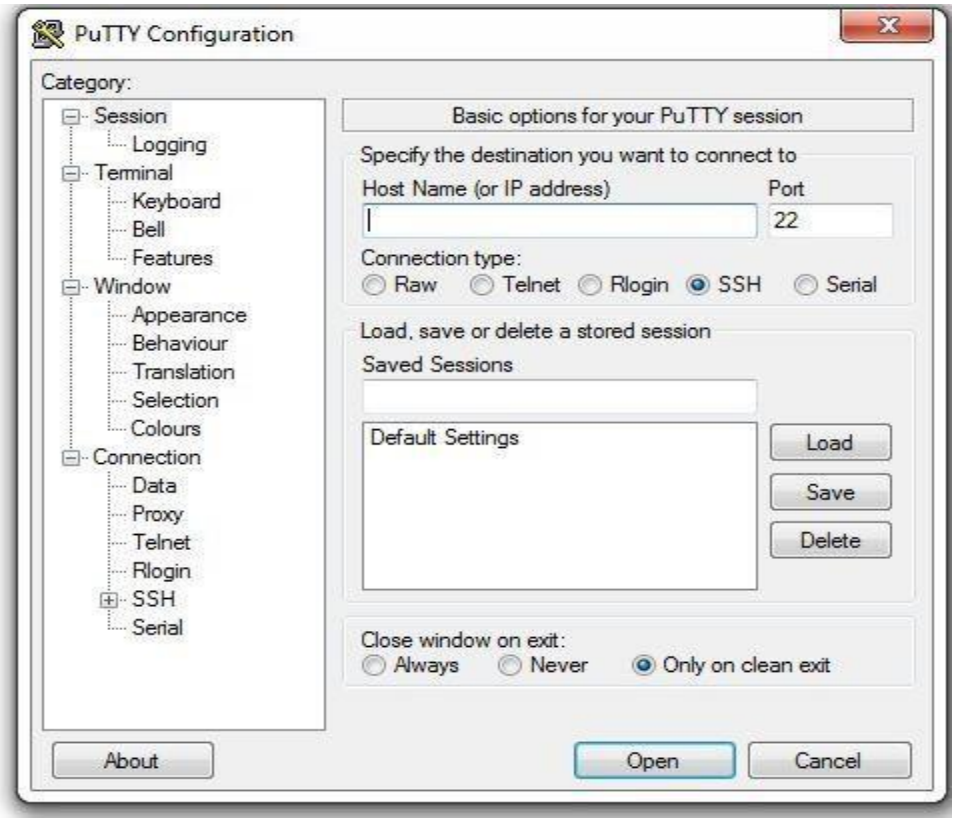
ram ram disk device
rar Archive files with compression
rcp Copy files between two machines
read Read a line from standard input •
readarray Read from stdin into an array variable •
readonly Mark variables/functions as readonly
reboot Reboot the system
rename Rename files
renice Alter priority of running processes
remsync Synchronize remote files via email
return Exit a shell function
rev Reverse lines of a file
rm Remove files

```
  rmdir    Remove folder(s)

s
  screen   Multiplex terminal, run remote shells via ssh
  scp      Secure copy (remote file copy)
  sdiff    Merge two files interactively
  sed      Stream Editor
  select   Accept keyboard input
  seq      Print numeric sequences
  set      Manipulate shell variables and functions
  sftp     Secure File Transfer Program
  shift    Shift positional parameters
  shopt    Shell Options
  shutdown Shutdown or restart linux
  sleep    Delay for a specified time
  slocate  Find files
  sort     Sort text files
  source   Run commands from a file '.'
  split    Split a file into fixed-size pieces
  ss       Socket Statistics
  ssh      Secure Shell client (remote login program)
  su       Substitute user identity
  sudo     Execute a command as another user
  sum      Print a checksum for a file
  suspend  Suspend execution of this shell •

t
  tail     Output the last part of file
  tar      Store, list or extract files in an archive
  tee      Redirect output to multiple files
  test     Evaluate a conditional expression
  time     Measure Program running time
  timeout  Run a command with a time limit
  times    User and system times
  touch    Change file timestamps
  top      List processes running on the system
  tput     Set terminal-dependent capabilities, color, position
  traceroute Trace Route to Host
  trap      Execute a command when the shell receives a signal •
  tr       Translate, squeeze, and/or delete characters
  true     Do nothing, successfully
  tsort    Topological sort
  tty      Print filename of terminal on stdin
  type     Describe a command •
u
  ulimit   Limit user resources •
  umask    Users file creation mask
  umount   Unmount a device
```

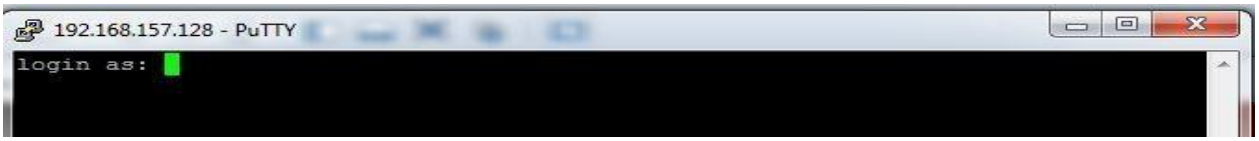
```
  unalias  Remove an alias •
  uniq     Uniquify files
  units    Convert units from one scale to another
  until    Execute commands (until error)
  uptime   Show uptime
  useradd  Create new user account
  userdel  Delete a user account
  usermod  Modify user account
  users    List users currently logged in

v
  v        Verbosely list directory contents (`ls -l -b')
  vdir     Verbosely list directory contents (`ls -l -b')
  vi       Text Editor
  vmstat   Report virtual memory statistics
w
  w        Show who is logged on and what they are doing
  wait     Wait for a process to complete •
  watch    Execute/display a program periodically
  wc       Print byte, word, and line counts
  whereis  Search the user's $path, man pages and source files
for a program
  which    Search the user's $path for a program file
  while    Execute commands
  who      Print all usernames currently logged in
  whoami   Print the current user id and name (`id -un')
  wget     Retrieve web pages or files via HTTP, HTTPS or FTP
  write    Send a message to another user
x
  xargs    Execute utility, passing constructed argument list(s)
  xdg-open Open a file or URL in the user's preferred application.
  xz       Compress or decompress .xz and .lzma files
  yes      Print a string until interrupted
  zip      Package and compress (archive) files.
  .        Run a command script in the current shell
  !!       Run the last command again
  ###      Comment / Remark
```

Procedure to connect to LINUX(putty)

Step 1:click on putty icon available on desk top. A window is opened



Step 2:fill in ip address of linux server and click open



Step 3: provide login and password (nothing is displayed on screen while typing password)

Step 4: change the default password at your first login

EXPERIMENT NO: 1 Date:

Practice File handling utilities, Process utilities, Disk utilities, Networking commands, Filters, Text processing utilities and Backup utilities.

Aim: Practice basic commands of Linux

File handling utilities

Cat Command: cat linux command concatenates files and print it on the standard output.

To Create a new file:
cat > file1.txt
This command creates a new file file1.txt. After typing into the file press control+d (^d) simultaneously to end the file.

To Append data into the file: To append data into the same file use append operator >> to write into the file, else the file will be overwritten (i.e., all of its contents will be erased).
cat >> file1.txt

To display a file: This command displays the data in the file.
cat file1.txt

To concatenate several files and display:
cat file1.txt file2.txt

The above cat command will concatenate the two files (file1.txt and file2.txt) and it will display the output in the screen. Some times the output may not fit the monitor screen. In such situation you can print those files in a new file or display the file using less command.

cat file1.txt file2.txt | less

To concatenate several files and to transfer the output to anotherfile.
cat file1.txt file2.txt > file3.txt

In the above example the output is redirected to new file file3.txt.

rm COMMAND:
rm linux command is used to remove/delete the file from the directory.

To Remove / Delete a file: Here rm command will remove/delete the file file1.txt.
rm file1.txt

To delete a directory tree:
rm -ir tmp

This rm command recursively removes the contents of all subdirectories of the tmp directory, prompting you regarding the removal of each file, and then removes the tmpdirectory itself.

To remove more files at once: rm command removes file1.txt and file2.txt files at the same time.
rm file1.txt file2.txt

cd COMMAND: cd command is used to change the directory.

cd linux-command
This command will take you to the sub-directory(linux-command) from its parent directory.
Ex:

cd ..
This will change to the parent-directory from the current working directory/sub-directory.

cd ~
This command will move to the user's home directory which is "/home/username".

cp COMMAND:
cp command copy files from one location to another. If the destination is an existing file, then the file is overwritten; if the destination is an existing directory, the file is copied into the directory (the directory is not overwritten).

Copy two files:
cp file1.txt file2.txt
The above cp command copies the content of file1.txt to file2.txt
Ex:

ls COMMAND:
ls command lists the files and directories under current working directory. Display root directory contents:

ls /
lists the contents of root directory.

Display hidden files and directories:
ls -a
lists all entries including hidden files and directories.

Display inode information:
ls –i

ln COMMAND:
ln command is used to create link to a file (or) directory. It helps to provide soft link for desired files.

Inode will be different for source and destination.
ln -s file1.txt file2.txt

Creates a symbolic link to 'file1.txt' with the name of 'file2.txt'. Here inode for 'file1.txt' and 'file2.txt' will be different.

mkdir command:

rmdir command:

mv command:

diff command:

comm command:

wc command:

Process utilities:

ps Command:
ps command is used to report the process status. ps is the short name for Process Status.

1. ps: List the current running processes.
Output:
PID TTY TIME CMD
2540 pts/1 00:00:00 bash

2. ps –f : Displays full information about currently running processes.
Output:
UID PID PPID C STIME TTY TIME CMD

nirmala 2540 2536 0 15:31 pts/1 00:00:00 bash

3. kill COMMAND: kill command is used to kill the background process.

Step by Step process:

• Open a process music player or any file.

xmms

press ctrl+z to stop the process.

• To know group id or job id of the background task.

jobs -l

It will list the background jobs with its job id as,

• xmms 3956

• kmail 3467

To kill a job or process.

• kill 3956

kill command kills or terminates the background process xmms.

Disk utilities:

du (abbreviated from disk usage) is a standard Unix program used to estimate file space usage—space used under a particular directory or files on a file system.

$du kt.txt pt.txt /* the first column displayed the file's disk usage */

8 kt.txt

4 pt.txt

Using -h option: As mentioned above, -h option is used to produce the output in human readable format.

$du -h kt.txt pt.txt

8.0K kt.txt

4.0K pt.txt

/*now the output is in human readable format i.e in Kilobytes */

Using -a option

$du -a kartik

8 kartik/kt.txt

4 kartik/pt.txt

4 kartik/pranjal.png

4 kartik/thakral.png

4 kartik/thakral

24 kartik

/*so with -a option used all the files (under directory kartik) disk usage info is displayed along with the thakral sub-directory */

df command : Report file system disk space usage

$df kt.txt

Filesystem	1K-blocks	Used	Available	Use%	Mounted on
/dev/the2	1957124	1512	1955612	1%	/snap/core

/* the df only showed the disk usage details of the file system that contains file kt.txt */

//using df without any filename //

$df

/* in this case df displayed the disk usage details of all mounted file systems */

Using -h : This is used to make df command display the output in human-readable format.

//using -h with df//

$df -h kt.txt

Filesystem 1K-blocks Used Available Use% Mounted on
/dev/the2 1.9G 1.5M 1.9G 1% /snap/core
/*this output is easily understandable by the user and all cause of -h option */

Networking commands

ping

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

Syntax: $ping hostname or ip-address

The above command starts printing a response after every second. To come out of the command, you can terminate it by pressing CNTRL + C keys.

$ping google.com

PING google.com (74.125.67.100) 56(84) bytes of data.
64 bytes from 74.125.67.100: icmp_seq=1 ttl=54 time=39.4 ms

ftp: ftp stands for File Transfer Protocol. This utility helps you upload and download your file from one computer to another computer.

Syntax $ftp hostname or ip-address

$ftp amrood.com

Connected to amrood.com.
220 amrood.com FTP server (Ver 4.9 Thu Sep 2 20:35:07 CDT 2009)
Name (amrood.com:amrood): amrood
331 Password required for amrood.
Password:
230 User amrood logged in.
ftp> dir
200 PORT command successful.
….
ftp> quit
221 Goodbye.

telnet:

Telnet is a utility that allows a computer user at one site to make a connection, login and then conduct work on a computer at another site. Once you login using Telnet, you can perform all the activities on your remotely connected machine.

C:>telnet amrood.com
Trying...
Connected to amrood.com.
Escape character is '^]'.
login: amrood
amrood's Password:
****
WELCOME TO AMROOD.COM *
********** $ logout

Connection closed.
C:>

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

Check all the logged-in users on the local machine −

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

Check all the logged-in users on the remote machine –

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

Get the information about a specific user available on the remote machine −

```
$ finger amrood@avtar.com
```

Ifconfig: Ifconfig is used to configure the network interfaces.

Filters

more COMMAND:
more command is used to display text in the terminal screen. It allows only backward movement.

1. more -c index.txt
Clears the screen before printing the file .

2. more -3 index.txt
Prints first three lines of the given file. Press Enter to display the file line by line.

head COMMAND:
head command is used to display the first ten lines of a file, and also specifies how many lines to display.

1. head index.php
This command prints the first 10 lines of 'index.php'.
2. head -5 index.php
The head command displays the first 5 lines of 'index.php'.
3. head -c 5 index.php
The above command displays the first 5 characters of 'index.php'.

tail COMMAND:
tail command is used to display the last or bottom part of the file. By default it displays last 10 lines of a file.

1. tail index.php
It displays the last 10 lines of 'index.php'.
2. tail -2 index.php
It displays the last 2 lines of 'index.php'.
3. tail -n 5 index.php

It displays the last 5 lines of 'index.php'.
4. tail -c 5 index.php
It displays the last 5 characters of 'index.php'.

cut COMMAND:
cut command is used to cut out selected fields of each line of a file. The cut command uses delimiters to determine where to split fields.

cut -c1-3 text.txt
Output:
Thi
Cut the first three letters from the above line.

paste COMMAND:
paste command is used to paste the content from one file to another file. It is also used to set column format for each line.

paste test.txt>test1.txt
Paste the content from 'test.txt' file to 'test1.txt' file.

sort COMMAND:
sort command is used to sort the lines in a text file.

1. sort test.txt
Sorts the 'test.txt'file and prints result in the screen.
2. sort -r test.txt
Sorts the 'test.txt' file in reverse order and prints result in the screen.

uniq
Report or filter out repeated lines in a file.

uniq myfile1.txt > myfile2.txt - Removes duplicate lines in the first file1.txt and outputs the results to the second file.

Text processing utilities

echo: display a line of text or echo command prints the given input string to standard output.
eg. echo I love India
echo $HOME

wc: print the number of newlines, words, and bytes in files
eg. wc file1.txt

nl: which lets you number lines in files.
eg. $ nl file1
1 hi
join- Join command is used for merging the lines of different sorted files based on the presence of common field into a single line. The second line will be appended at the end of the first line and cursor is placed at the end of line after joining.

$cat file1.txt	**$cat file2.txt**	**$join file1.txt file2.txt**
1 AAYUSH	1 101	
2 APAAR	2 102	
3 HEMANT	3 103	
4 KARTIK	4 104	

Grep (Global Regular Expression Searching for a pattern), fgrep and egrep
$ grep "sales director" emp1 emp2

$fgrep 'good bad great' userfile
$egrep 'good | bad | great' userfile

cat, head, tail, sort, uniq, cut, paste and etc.

Backup utilities

Linux backup and restore can be done using backup commands tar, cpio, dump and restore.

Backup Restore using tar command

tar: tape archive is used for single or multiple files backup and restore on/from a tape or file.
$tar cvf /dev/rmt/0 *

Options: c -> create ; v -> Verbose ; f->file or archive device ; * -> all files and directories .

$tar cvf /home/backup *
Create a tar called backup in home directory, from all file and directories s in the current directory.

Viewing a tar backup on a tape or file
$tar tvf /dev/rmt/0 ## view files backed up on a tape device.
$tar tvf /home/backup ## view files backed up inside the backup
Note: t option is used to see the table of content in a tar file.

Extracting tar backup from the tape
$tar xvf /home/backup ## extract / restore files in to current directory.
Note : x option is used to extract the files from tar file. Restoration will go to present directory or original backup path depending on relative or absolute path names used for backup.

Backup restore using cpio command

Using cpio command to backup all the files in current directory to tape.
find . -depth -print | cpio -ovcB > /dev/rmt/0

cpio expects a list of files and find command provides the list, cpio has to put these file on some destination and a > sign redirect these files to tape. This can be a file as well .

Viewing cpio files on a tape

cpio -ivtB < /dev/rmt/0
Options i -> input ; v->verbose; t-table of content; B-> set I/O block size to 5120 bytes

Restoring a cpio backup
cpio -ivcB < /dev/rmt/0
Options i -> input ; v->verbose; t-table of content; B-> set I/O block size to 5120 bytes

Signature of the Faculty

EXPERIMENT NO: 2a Date:

Aim:-Write a shell script that receives any number of file names as arguments checks if every argument supplied is a file or directory and reports accordingly. Whenever the argument is a file it reports no of lines present in it

ALGORITHM:

step 1: if arguments are less than 1 print Enter at least one input file name and goto step 9
Step 2: selects list a file from list of arguments provided in command line
Step 3: check for whether it is directory if yes print is directory and goto step 9
step 4: check for whether it is a regular file if yes goto step 5 else goto step 8
Step 5: print given name is regular file
step 6: print No of lines in file
step 7: goto step
step 8: print not a file or a directory
step 9: stop

Script name: 2a.sh

```
#!/bin/bash
if [ $# -lt 1 ]
then
	echo "Enter at least one input file name"
else
	for i in $*
	do
		if [ -d $i ]
		then
			echo " $i is directory" elif [ -f $i ]
		then
			echo " given name is file: $i"
			echo " No of lines in file are : `wc -l $i`"
		else
			echo "given name is not a file or a directory"
		fi
	done
fi
```

Execution:
provide two file names as input one a regular file and other directory
for example abc1.txt a text file as first argument and mrcet a directory as second argument

Run1:

```
[root@localhost sh]# sh 2a.sh abc1.txt mrcet
given name is file: abc1.txt
No of lines in file are : 7 abc1.txt
mrcet is directory
```

run 2:[root@localhost sh]# sh 2a.sh abc1.txt abc2.txt

given name is file: abc1.txt
No of lines in file are : 7 abc1.txt
given name is file: abc2.txt
No of lines in file are : 7 abc2.txt

Viva Questions:

1. What is an internal command in Linux?
Internal commands are also called shell built-in commands. Example: cd,fg. Since these are shell built-in, no process is created while executing these commands, and hence are considered to be much faster.

2. x and y are two variables containing numbers? How to add these 2 numbers?
$ expr $x + $y

3. How to add a header record to a file in Linux?
$ sed -i '1i HEADER' file

4. How to find the list of files modified in the last 30 mins in Linux?
$ find . -mmin -30

5. How to find the list of files modified in the last 20 days?
$ find . -mtime -20

Assignment :-

Sno	Task	Date	Sign	Remark
1	Write a shell script to count no of regular files in the current working directory			
2	Write a shell script to display list of currently logged users			
3	Write a Shell Script that accepts a file name, starting and ending line numbers as Arguments and displays all lines between the given line numbers.			
4	Write a shell script to perform arithmetic operation using case statement			

Signature of the Faculty

EXPERIMENT NO: 2b Date:

Aim:-Write a shell script that accepts a list of file names as its arguments, counts and reports the occurrence of each word that is present in the first argument file on other argument files.

ALGORITHM:
step1: Check the no of arguments for shell script
if 0 arguments then print no arguments
step2:else translate each word in the first file is to be on separate line
which will be stored in temp file
step3: for i in $*
for every filename in given files
step 4: translate each word in the file is to be on separate line
which will be stored in temp1 file
step5: count no of lines in temp file assign it to j
step6: initialize j=1
step 7: while i< j
extract the line that are common in both the file by using
head and tail commands
then apply the filter grep to count and print the lines
which are common to files
increment j
step 8: stop

Script name: 2b.sh

```
#!/bin/bash
echo "no of arguments $#"
if [ $# -le 2 ]
then
        echo "Error : Invalid number of arguments."
        exit
fi
str=`cat $1 | tr '\n' ' '`
for a in $str
do
        echo "in file $a"
        echo "Word = $a, Count = `grep -c "$a" $2`"
done
```

Execution and output: check data in abc1.txt file

```
[root@localhost sh]# cat abc1.txt
abc
def
ghi
abc
abc
```

cccc
check data in abc1.txt file
[root@localhost sh]# cat abc2.txt
abc
def
ghi
abc
abc
cccc

executing script

[root@localhost sh]# sh 2b.sh abc1.txt
abc2.txt Word = abc, Count = 3
Word = def, Count = 1
Word = ghi, Count = 1
Word = abc, Count = 3
Word = abc, Count = 3
Word = cccc, Count = 1

Viva Questions

1. What is Shell Scripting ?
Shell scripting, in Linux or Unix, is programming with the shell using which you can automate your tasks. A shell is the command interpreter which is the interface between the User and the kernel. A shell script allows you to submit a set of commands to the kernel in a batch. In addition, the shell itself is very powerful with many properties on its own, be it for string manipulation or some basic programming stuff.
2. The command "cat file" gives error message "--bash: cat: Command not found". Why?
It is because the PATH variable is corrupt or not set appropriately. And hence the error because the cat command is not available in the directories present PATH variable.
3. How to find the length of a string in Linux?
$ x="welcome" $ echo ${#x} 7
4. What are the different timestamps associated with a file?
Modification time:- Refers to the time when the file is last modified.
Access time :- The time when the file is last accessed.
Changed time :- The time when the attributes of the file are last changed.

5. How to get the list of files alone in a directory in Linux?
$ ls -lrt | grep ^-

Assignment :-

Sno	Task	Date	Sign	Remark
1	Write a shell script to print prime numbers			
2	Write a shell script to print Fibonacci numbers			

Signature of the Faculty

EXPERIMENT NO: 3a Date:

Aim:-Write a shell script to list all of the directory files in a directory.

Algorithm:

Step1: enter the name of the directory
Read dir
Step2: if it is a directory
Then list the files present in that directory
By using ls command with –p option to list all directory files in a given directory
Step 3: else enter the directory name
Step 4: stop

Script name: 3a.sh

```
#!/bin/bash
echo " Enter dir name: "
read dir
if [ -d $dir ]
then
	printf " Files in Directory $dir are...\n`ls $dir`"
else
	echo " Dir does not exist"
fi
```

Execution and output:

```
[root@localhost sh]# sh 3a.sh
Enter dir name:
japs
Files in Directory japs are...
abc1.txt
abc2.txt
ls-l.c
prg5
s1
```

Viva Questions

1. A string contains a absolute path of a file. How to extract the filename alone from the absolute path in Linux?

```
$ x="/home/guru/temp/f1.txt"
$ echo $x | sed 's^.*/^^'
```

2. How to find all the files created after a pre-defined date time, say after 10th April 10AM?

This can be achieved in 2 steps:
1. Create a dummy file with the time stamp, 10th April 10AM.
2.Find all the files created after this dummy file.

```
$ touch -t 1004101000 file
```

$ find . -newer file

3. The word "Unix" is present in many .txt files which is present across many files and also files present in sub directories. How to get the total count of the word "Unix" from all the .txt files?

$ find . -name *.txt -exec grep -c Unix '{}' \; | awk '{x+=$0;}END{print x}'

Assignment :-

Sno	Task	Date	Sign	Remark
1	How to find the files modified exactly before 30minutes? $ find . -mmin 30			
2	How to print the contents of a file line by line in Linux?			

Signature of the Faculty

EXPERIMENT NO: 3b Date:

AIM: Write a shell script that deletes all lines containing the specified word in one or more files Supplied as arguments to it.

ALGORITHM:

Step 1: Create a file with 5-6 lines of data
Create the 2 file f1 and f2 as vi s1and vi s2
Step2: Now write a shell script with
vi 2.sh
step3:Check the no of arguments for shell script
if 0 arguments then print no arguments
else pattern=$1(word will be stored in pattern)
for fname in $*
for every filename in given files
if it is a file if [-f $fname] then
print DELETING $pattern FROM
$fname sed '/'$pattern'/d' $fname
sed acts as filter if word is a file in any line that will be deleted
'/' is used to represent regular expressions
'/d' is a delete command in sed
else print file NOT FOUND

Script name: 3b.sh

```
#!/bin/bash
if [ $# -lt 2 ]then
        echo "Enter atlest two files as input in command line"
else
        printf "enter a word to find:"
        read word
        for f in $*
        do
                printf "\n In File $f:\n"
                sed /$word/d $f
        done
fi
```

Execution:
run1:
check data in input files

```
[root@localhost sh]# cat abc1.txt
abc
def
ghi
abc
abc
cccc
[root@localhost sh]# cat abc2.txt
abc
```

```
def
ghi
abc
abc
cccc
Executing shell script
[root@localhost sh]# sh 3b.sh abc1.txt abc2.txt
enter a word to find:abc
In File abc1.txt:
def
ghi
cccc
In File abc2.txt:
def
ghi
cccc
```

Expected output:
Displays lines from files s1 s2 after deleting the word hi

Viva Questions

1. Explain various loops in shell script
2.Explain grep
3. Explain egrep
4. Explain fgrep
5. .Explain sed

Assignment:-

Sno	Task	Date	Sign	Remark
1	Write a shell script to count occurrence of a word in a file			
2	Write a shell script to print line numbers in which a particular word has occurred where word is provides as input.			
3	Write a shell script that displays a list of all files in the current directory to which the user has read, write and execute permissions.			

Signature of the Faculty

EXPERIMENT NO: 3c Date:

Aim:-Write a shell script to find factorial of a given number.

ALGORITHM

Step 1: read any number to find factorial
Step 2: initialize fact=1 and i=1
Step 3: while i less than
do
fact=fact* i
i=i+1
done
step 4:print fact
step 5:stop.

Script Name:3c.sh

```
#!/bin/bash
echo "Factorial Calculation Script. .."
echo "Enter a number: "
read f
fact=1
factorial=1
while [ $fact -le $f ]
do
        factorial=`expr $factorial \* $fact`
        fact=`expr $fact + 1`
done
echo "Factorial of $f = $factorial"
```

Execution and Output:

```
[root@localhost sh]# sh 3c.sh
Factorial Calculation Script....
Enter a number: 4
Factorial of 4 = 24
```

Assignment :-

Sno	Task	Date	Sign	Remark
1	Write a shell script to find sum of first n natural numbers			
2	Write a shell script to find largest of given three numbers			

Signature of the Faculty

EXPERIMENT NO: 4a Date:

Aim:-write an awk script to count number of lines in a file that does not contain vowels

ALGORITHM

Step 1: create a file with 5-10 lines of data

Step 2: write an awk script by using grep command to filter the lines
that do not contain vowels
awk ‘ $0 ~/aeiou/ {print $0}’ file1

step3: count=count+1

step4:print count

step5:stop

Awk script name:nm.awk

```
BEGIN{}
{
If($0 !~/[aeiou AEIOU]/)
wordcount+=NF
}
END
{
print "Number of Lines are", wordcount
}
```

input file for awk script:data.dat

```
bcdfghj
abcdfghj
bcdfghj
ebcdfghj
bcdfghj
ibcdfghj
bcdfghj
obcdfghj
bcdfghj
ubcdfghj
```

Executing the script:

[root@localhost awk]# **awk -f nm.awk data.dat**

```
bcdfghj
bcdfghj
bcdfghj
bcdfghj
bcdfghj
```

Number f lines are 5

Assignment :-

Sno	Task	Date	Sign	Remark
1	Write an awk script to find square root of a given number			
2	Write an awk script to find maximum of two numbers , read input from keyboard			

Signature of the Faculty

EXPERIMENT NO: 4b Date:

Aim:-write an awk script to find the no of characters ,words and lines in a file

ALGORITHM

Step 1: create a file with 5 to10 lines of data
Step 2: write an awk script
find the length of file
store it in chrcnt
step3: count the no of fields (NF), store it in wordcount
step4: count the no of records (NR), store it in NR
step5: print chrcnt,NRwordcount
step6: stop

Awk script name:nc.awk

```
BEGIN{}
{
        print len=length($0),"\t",$0
        wordcount+=NF
chrcnt+=len
        }
END {
        print "total characters",chrcnt
        print "Number of Lines are",NR
        print "No of Words count:",wordcount
}
```

input data file name:data.dat

```
bcdfghj
abcdfghj
bcdfghj
ebcdfghj
bcdfghj
ibcdfghj
bcdfghj
obcdfghj
bcdfghj
ubcdfghj
```

Executing the script:
[root@localhost awk]# **awk -f nc.awk data.dat**

```
1 bcdfghj
2 abcdfghj
7 bcdfghj
8 ebcdfghj
7 bcdfghj
```

7 bcdfghj
8 obcdfghj
7 bcdfghj
8 ubcdfghj
total characters 75
Number of Lines are 10
No of Words count: 10

VIVA QUESTIONS:

1. How to find the last modified file or the newest file in a directory?
$ ls -lrt | grep ^- | awk 'END{print $NF}'

2. How to access the 10th command line argument in a shell script in Linux?
$1 for 1st argument, $2 for 2nd, etc... For 10th argument, ${10}, for 11th, ${11} and so on.

3. How to find the sum of all numbers in a file in Linux?
$ awk '{x+=$0}END{print x}' file

4. How to delete a file which has some hidden characters in the file name?
Since the rm command may not be able to delete it, the easiest way to delete a file with some hidden characters in its name is to delete it with the find command using the inode number of the file.
$ ls –li
total 32
9962571 -rw-r--r-- 1 guru users 0 Apr 23 11:35
$ find . -inum 9962571 -exec rm '{}' \;

5. Using the grep command, how can you display or print the entire file contents?
$ grep '.*' file

6. What is the difference between a local variable and environment variable in Linux?

A local variable is the one in which the scope of the variable is only in the shell in which it is defined. An environment variable has scope in all the shells invoked by the shell in which it is defined.

Signature of the Faculty

EXPERIMENT NO: 5 Date:

Aim: Implement in c language the following Unix commands using system calls
a)cat b)ls c) Scanning Directories (Ex: opendir(), readdir(),etc.)

a) AIM:-Write a c program to implement **cat command** using system calls

Description:

cat COMMAND: cat linux command concatenates files and print it on the standard output.

SYNTAX:

cat [OPTIONS] [FILE]...

OPTIONS:

-A Show all.
-b Omits line numbers for blank space in the output.
-e A $ character will be printed at the end of each line prior to a new line.
-E Displays a $ (dollar sign) at the end of each line.
-n Line numbers for all the output lines.
-s If the output has multiple empty lines it replaces it with one empty line.
-T Displays the tab characters in the output.
-v Non-printing characters (with the exception of tabs, new-lines & form-feeds) are printed visibly.

Operations With cat Command:

1. To Create a new file:

$cat > file1.txt
This command creates a new file file1.txt. After typing into the file press control+d (^d) simultaneously to end the file.

2. To Append data into the file:
$cat >> file1.txt
To append data into the same file use append operator >> to write into the file, else the file will be overwritten (i.e., all of its contents will be erased).

3. To display a file:
$cat file1.txt
This command displays the data in the file.

4. To concatenate several files and display:

$cat file1.txt file2.txt
The above cat command will concatenate the two files (file1.txt and file2.txt) and it will display the output in the screen. Some times the output may not fit the monitor screen. In such situation you can print those files in a new file or display the file using less command.
cat file1.txt file2.txt | less

5. To concatenate several files and to transfer the output to another file.
$cat file1.txt file2.txt > file3.txt
In the above example the output is redirected to new file file3.txt. The cat command will create new file file3.txt and store the concatenated output into file3.txt.

Algorithm:

Step 1:Start
Step 2:read arguments from keyboard at command line
Step 3:if no of arguments are less than two print ENTER CORRECT ARGUMENTS
Else goto step 4
Step4:read the date from specified file and write it to destination file
Step 5 :stop

Program file name: 5a.c

```
#include<stdio.h>
        #include<sys/types.h>
        #include<stdlib.h>
        #include<fcntl.h>
        #include<sys/stat.h>
        int main(int argc,char *argv[])
        {
        int fd,n;
        char buff[512];
                if(argc!=2)
                        printf("ENTER CORRECT ARGUMENTS :");
                if((fd=open(argv[1],4))<0)
                {
                        printf("ERROR");
                                return 0;
                }
                while(n=read(fd,buff,sizeof(buff))>0)
                        write(1,buff,n);
        }
```

b) AIM:-Write a c program to implement **ls command** using system calls

Description:

ls command is used to list the files present in a directory

Algorithm:

Step 1. Start.
Step 2. open directory using opendir( ) system call.
Step 3. read the directory using readdir( ) system call.
Step 4. print dp.name and dp.inode .
Step 5. repeat above step until end of directory.
Step 6: Stop.

Program name: 5b.c

```
#include<stdio.h>
#include<dirent.h>
void quit(char*,int);
int main(int argc,char **argv )
{
```

```
        DIR *dirop;
        struct dirent *dired;
        if(argc!=2)
        {
                printf("Invalid number of arguments\n");
        }
        if((dirop=opendir(argv[1]))==NULL)
                printf("Cannot open directory\n");
        while((dired=readdir(dirop))!=NULL)
                printf("%10d %s\n",dired>d_ino,dired>d_name);
        closedir(dirop);
}
```

c) **Aim :** write a c program that simulates **Scanning directories** (using system calls)

Description:

Scanning directories is used to opendir(), readdir(), rewinddir(), closedir(), etc.)

Algorithm:

Program File name: 5c.c

Expected Output:

Signature of the Faculty

EXPERIMENT NO: 6 Date:

Aim: Write a C program that takes one or more file/directory names as command line input and reports following information

A) File Type B)Number Of Links
C) Time oflast Acces D) Read,write and execute permissions

Algorithm:

Step 1:start
Step 2:Declare struct stat a
Step 3:read arguments at command line
Step 4: set the status of the argument using stat(argv[i],&a);
Step 5:Check whether the given file is Directory file by using S_ISDIR(a.st_mode)
if it is a directory file print Directory file
Else
print is Regular file
Step6: print number of links
Step 7:print last time access
Step 8:Print Read,write and execute permissions
Step 9:stop

Program File name: 6.c

```
#include<stdio.h>
#include<sys/stat.h>
#include<time.h>
int main(int argc,char *argv[])
{
int  i,j;
struct stat a;
for (i=1;i<argc;i++)
{
        printf("%s : ",argv[i]);
        stat(argv[i],&a);
        if(S_ISDIR(a.st_mode))
        {
                printf("is a Directory file\n");
        }
        else
        {
                printf("is Regular file\n");
        }
        printf("******File Properties********\n");
        printf("Inode Number:%d\n",a.st_ino);
        printf("UID:%o\n",a.st_uid);
        printf("GID:%o\n",a.st_gid);
        printf("No of Links:%d\n",a.st_nlink);
        printf("Last Access time:%s",asctime(localtime(&a.st_atime)));
```

```
        printf("Permission flag:%o\n",a.st_mode%512);
        printf("size in bytes:%d\n",a.st_size);
        printf("Blocks Allocated:%d\n",a.st_blocks);
        printf("Last modification time %s\n",ctime(&a.st_atime));
        }
}
```

Assignment:

Sno	Task	Date	Sign	Remark
1	write a c program that simulates mkdir command using system calls			
2	write a c program that simulates rmdir command using system calls			

Signature of the Faculty

EXPERIMENT NO: 7a Date:

Write a C program to implement kill(), raise() and sleep() functions.

Aim: Implement kill(), raise() and sleep() functions using a C program.

kill( ) and sleep( ):

Program file name: kill.c

```
#include <signal.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>

// function declaration
void sighup();
void sigint();
void sigquit();

// driver code
void main()
{
    int pid;

    /* get child process */
    if ((pid = fork()) < 0) {
        perror("fork");
        exit(1);
    }

    if (pid == 0) { /* child */
        signal(SIGHUP, sighup);
        signal(SIGINT, sigint);
        signal(SIGQUIT, sigquit);
        for (;;)
            ; /* loop for ever */
    }

    else /* parent */
    { /* pid hold id of child */
        printf("\nPARENT: sending SIGHUP\n\n");
        kill(pid, SIGHUP);

        sleep(3); /* pause for 3 secs */
        printf("\nPARENT: sending SIGINT\n\n");
        kill(pid, SIGINT);
          sleep(3); /* pause for 3 secs */
```

```
printf("\nPARENT: sending SIGQUIT\n\n");
        kill(pid, SIGQUIT);
        sleep(3);
    }
}
  // sighup() function definition
void sighup()
  {
    signal(SIGHUP, sighup); /* reset signal */
    printf("CHILD: I have received a SIGHUP\n");
}

// sigint() function definition
void sigint()
  {
    signal(SIGINT, sigint); /* reset signal */
    printf("CHILD: I have received a SIGINT\n");
}
  // sigquit() function definition
void sigquit()
{
    printf("My DADDY has Killed me!!!\n");
    exit(0);
}
```

OP: $ ./a.out

```
PARENT: sending SIGHUP
CHILD: I have received a SIGHUP
PARENT: sending SIGINT
CHILD: I have received a SIGINT
PARENT: sending SIGQUIT
My Parent has Killed me!!!
```

raise( ):

Program file name: raise.c

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

void signal_handler(int signal)
{
    /* Display a message indicating we have received a signal */
    if (signal == SIGUSR1) printf("Received a SIGUSR1 signal\n");

/* Exit the application */
exit(0);
}
```

```
int main(int argc, const char * argv[])
{
    /* Display a message indicating we are registering the signal handler */
    printf("Registering the signal handler\n");

    /* Register the signal handler */
    signal(SIGUSR1, signal_handler);

    /* Display a message indicating we are raising a signal */
    printf("Raising a SIGUSR1 signal\n");

    /* Raise the SIGUSR1 signal */
    raise(SIGUSR1);

    /* Display a message indicating we are leaving main */
    printf("Finished main\n");

    return 0;
}
```

Output:

Registering the signal handler
Raising a SIGUSR1 signal
Received a SIGUSR1 signal

Signature of the Faculty

EXPERIMENT NO: 7b Date:

Write a C program to implement alarm(), pause() and abort() functions.

Aim: Implement alarm(), pause() and abort() functions using a C program.

Program file name: alarmpause.c

```
#define _POSIX_SOURCE
#include <unistd.h>
#include <signal.h>
#include <stdio.h>
#include <time.h>

void catcher(int signum) {
  puts("inside catcher...");
}

void timestamp() {
  time_t t;
  time(&t);
  printf("the time is %s", ctime(&t));
}

main() {
  struct sigaction sigact;

  sigemptyset(&sigact.sa_mask);
  sigact.sa_flags = 0;
  sigact.sa_handler = catcher;
  sigaction(SIGALRM, &sigact, NULL);

  alarm(10);
  printf("before pause... ");
  timestamp();
  pause();
  printf("after pause... ");
  timestamp();
}
```

Output:

```
before pause... the time is Fri Jun 16 09:42:29 2001
inside catcher...

after pause... the time is Fri Jun 16 09:42:39 2001
```

abort():

```
/* abort.c -- terminates execution abnormally */
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
   abort();
   printf("\"abort() called prior to printf()\"\n");

   return 0;
}
```

Signature of the Faculty

EXPERIMENT NO: 8a **Date:**

Aim: Write a C program to create child process and allow parent process to display "parent" and the child to display "child" on the screen

Algorithm:

Step 1: start
Step2: call the fork() function to create a child
process fork function returns 2 values
step 3: which returns 0 to child process
step 4:which returns process id to the parent
process step 5:stop

Program file name: 8a.c

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
int main()
{
int pid,pid1,pid2;
        pid=fork();
        if(pid==-1)
        {
                printf("ERROR IN PROCESS CREATION \n");
                exit(0);
        }
        if(pid!=0)
        {
                pid1=getpid();
                printf("\n the parent process ID is %d", pid1);
        }
        else
        {

                pid2=getpid();
        }       printf("\n the child process ID is %d\n", pid2);
}
```

Output:

```
[root@dba ~]# cc -o 8 8a.c
[root@dba ~]# ./8

 the child process ID is
 4485 the parent process
 ID is 4484
```

Signature of the Faculty

EXPERIMENT NO: 8b Date:

Aim: Write a C program to create zombie process

Algorithm: Step 1:call fork function to create a child process
Step 2:if fork()>0
Then creation of Zombie
By applying sleep function for 10 seconds
Step 3: now terminate the child process
Step 4: exit status child process not reported to parent
Step 5: status any process which is zombie can known by
Applying ps(1) command
Step 6: stop

Program file name: 8b.c

```
#include <stdio.h>
#include<stdlib.h>
int main()
{
int pid;
        pid=fork();
        if(pid == 0)
        {       printf("Iam child my pid is %d\n",getpid());
                 printf("My parent pid is:%d\n",getppid());
                 exit(0);
        }
         else
        {       printf("I am parent, my pid is %d\n",getpid());
                sleep(100);
                exit(0);
         }
}
```

Execution:

To see zombie process, after running the program, open a new terminal Give this command $ps -el|grep a.out

First terminal

Compilation:

```
[root@dba ~]# cc 8b.c
```

Executing binary

```
[root@dba ~]# ./a.out
Iam child my pid is 4732
My parent pid is:4731
I am parent, my pid is 4731
```

Checking for zombie process. Z means zombie process

Second terminal

```
[root@dba ~]# ps -el|grep a.out
0 S     0  4731  4585  0  77   0 -   384 -     pts/3    00:00:00 a.out
1 Z     0  4732  4731  0  77   0 -     0 exit   pts/3    00:00:00 a.out <defunct>
```

EXPERIMENT NO: 8c Date:

Aim:-Write a C program to illustrate how an orphan process is created

Algorithm:

Step 1: call the fork function to create the child process
Step 2:if (pid==0)
Then print child id and parent id
else goto step 4
Step 3:Then sleep(10)
Print child id and parent id
Step 4: Print child id and parent id
Step 5:which gives the information of orphan process
Step 6:stop

Program file name:18.c

```
#include <stdio.h>
#include<stdlib.h>
int main()
 {int pid;
        printf("I am the original process with PID %d and PPID %d\n",getpid(),getppid());
        pid=fork();
        if(pid == 0)
        {       printf("I am child, my pid is %d ",getpid());
                printf("My Parent pid is:%d\n",getppid());
                sleep(10);
                 printf("Now my pid is %d ",getpid());
                printf("My parent pid is:%d\n",getppid());
                exit(0);
        }
        else
        {       sleep(10);
                printf("I am parent, my pid is %d\n",getpid());
                //printf("I am going to die\n");
        }
printf("PID:%d terminates...\n",getpid());
}
```

Execution:

Compilation : [root@dba ~]# cc -o 18 18-1.c

Executing Binary:

```
[root@dba ~]# ./18
I am the original process with PID 5960 and PPID 5778
I am child, my pid is 5961 My Parent pid is:5960
I am parent, my pid is 5960
PID:5960 terminates...
[root@dba ~]# Now my pid is 5961 My parent pid is:1
```

Assignment:

Sno	Task	Date	Sign	Remark
1	Write a program to illustrate Vfork();			
2	Write a program to illustrate fork();			

Signature of the Faculty

EXPERIMENT NO: 9a Date:

Aim:- Write a C program that illustrate communication between two process using unnamed pipes

Program file name: unnamed_pipe.c

```
#include<stdio.h>
#include<stdlib.h>
#include<sys/types.h>
#include<sys/stat.h>
#include<string.h>
#include<fcntl.h>
void server(int,int);
void  client(int,int);
int main()
{
int p1[2],p2[2],pid;
        pipe(p1);
        pipe(p2);
        pid=fork();
        if(pid==0)
        {
         close(p1[1]);
        close(p2[0]);
        server(p1[0],p2[1]);
        return 0;
 }
        close(p1[0]);
        close(p2[1]);
        client(p1[1],p2[0]);
        wait();
return 0;
}

void client(int wfd,int rfd)
{
int i,j,n;
 char fname[2000];
 char buff[2000];
 printf("ENTER THE FILE NAME :");
 scanf("%s",fname);
 printf("CLIENT SENDING THE REQUEST .... PLEASE WAIT\n");
sleep(10);
write(wfd,fname,2000);
n=read(rfd,buff,2000);
buff[n]='\0';
printf("THE RESULTS OF CLIENTS ARE.....\n");
write(1,buff,n);
}
```

```
void server(int rfd,int wfd)
 {
        int i,j,n;
       char fname[2000];
       char buff[2000];
        n=read(rfd,fname,2000);
        fname[n]='\0';
        int fd=open(fname,O_RDONLY);
        sleep(10);
        if(fd<0)
                write(wfd,"can't open",9);
        else
                 n=read(fd,buff,2000);
        write(wfd,buff,n);
}
```

Assignment:

Sno	Task	Date	Sign	Remark
1	Write a program to demonstrate the function of a pipe			
2	Write a program to demonstrate the pipe function using dup() system call			

EXPERIMENT NO: 9b Date:

Aim:- Write a C program that illustrate communication between two process using named pipes or FIFO

Algorithm:

Create two processes, one is fifoserver_twoway and another one is fifoclient_twoway.

Algorithm for fifoserver_twoway :

step 1:Start
step 2: Creates a named pipe (using library function mkfifo())
with name "fifo_twoway" in /tmp directory, if not
created.
step 3: Opens the named pipe for read and write purposes.
step 4: Here, created FIFO with permissions of read and write for Owner. Read for Group and no permissions for Others.
step 5: Waits infinitely for a message from the client.
step 6: If the message received from the client is not "end", prints the message and reverses the string. The reversed string is sent back to the client. If the message is "end", closes the fifo and ends the process.
step 7:stop.

Algorithm for client :

Step 1: start
Step 2: Opens the named pipe for read and write purposes.
Step 3: Accepts string from the user.
Step 4: Checks, if the user enters "end" or other than "end". Either way, it sends a message to the server. However, if the string is "end", this closes the FIFO and also ends the process.
Step 5: If the message is sent as not "end", it waits for the message (reversed string) from the client and prints the reversed string.
Step 6: Repeats infinitely until the user enters the string "end".
Step 7: stop

Programs:

```
/* Filename: fifoserver_twoway.c */
#include <stdio.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <fcntl.h>
#include <unistd.h>
#include <string.h>

#define FIFO_FILE "/tmp/fifo_twoway"
void reverse_string(char *);
int main() {
  int fd;
```

```
 char readbuf[80];
   char end[10];
   int to_end;
   int read_bytes;

   /* Create the FIFO if it does not exist */
   mkfifo(FIFO_FILE, S_IFIFO|0640);
   strcpy(end, "end");
   fd = open(FIFO_FILE, O_RDWR);
   while(1) {
      read_bytes = read(fd, readbuf, sizeof(readbuf));
      readbuf[read_bytes] = '\0';
      printf("FIFOSERVER: Received string: \"%s\" and length is %d\n", readbuf,
            (int)strlen(readbuf));
      to_end = strcmp(readbuf, end);

      if (to_end == 0) {
         close(fd);
         break;
      }
      reverse_string(readbuf);
      printf("FIFOSERVER: Sending Reversed String: \"%s\" and length is %d\n", readbuf, (int)
            strlen(readbuf));
      write(fd, readbuf, strlen(readbuf));
      /*
      sleep - This is to make sure other process reads this, otherwise this
      process would retrieve the message
      */
      sleep(2);
   }
   return 0;
}

void reverse_string(char *str) {
   int last, limit, first;
   char temp;
   last = strlen(str) - 1;
   limit = last/2;
   first = 0;

   while (first < last) {
      temp = str[first];
      str[first] = str[last];
      str[last] = temp;
      first++;
      last--;  }   return;}
```

Output:

```
FIFOSERVER: Received string: "LINUX IPCs" and length is 10
FIFOSERVER: Sending Reversed String: "sCPI XUNIL" and length is 10
FIFOSERVER: Received string: "Inter Process Communication" and length is 27
FIFOSERVER: Sending Reversed String: "noitacinummoC ssecorP retnI" and length is 27
FIFOSERVER: Received string: "end" and length is 3
```

/* Filename: fifoclient_twoway.c */

```
#include <stdio.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <fcntl.h>
#include <unistd.h>
#include <string.h>

#define FIFO_FILE "/tmp/fifo_twoway"
int main() {
  int fd;
  int end_process;
  int stringlen;
  int read_bytes;
  char readbuf[80];
  char end_str[5];
  printf("FIFO_CLIENT: Send messages, infinitely, to end enter \"end\"\n");
  fd = open(FIFO_FILE, O_CREAT|O_RDWR);
  strcpy(end_str, "end");

  while (1) {
    printf("Enter string: ");
    fgets(readbuf, sizeof(readbuf), stdin);
    stringlen = strlen(readbuf);
    readbuf[stringlen - 1] = '\0';
    end_process = strcmp(readbuf, end_str);

    //printf("end_process is %d\n", end_process);
    if (end_process != 0) {
      write(fd, readbuf, strlen(readbuf));
      printf("FIFOCLIENT: Sent string: \"%s\" and string length is %d\n", readbuf,
        (int)strlen(readbuf));
      read_bytes = read(fd, readbuf, sizeof(readbuf));
      readbuf[read_bytes] = '\0';
      printf("FIFOCLIENT: Received string: \"%s\" and length is %d\n", readbuf,
        (int)strlen(readbuf));
    } else {
      write(fd, readbuf, strlen(readbuf));
      printf("FIFOCLIENT: Sent string: \"%s\" and string length is %d\n", readbuf,
        (int)strlen(readbuf));
      close(fd);
```

```
      break;
      }
    }
    return 0;
}
```

Output:

```
FIFO_CLIENT: Send messages, infinitely, to end enter "end"
Enter string: LINUX IPCs
FIFOCLIENT: Sent string: "LINUX IPCs" and string length is 10
FIFOCLIENT: Received string: "sCPI XUNIL" and length is 10
Enter string: Inter Process Communication
FIFOCLIENT: Sent string: "Inter Process Communication" and string length is 27
FIFOCLIENT: Received string: "noitacinummoC ssecorP retnI" and length is 27
Enter string: end
FIFOCLIENT: Sent string: "end" and string length is 3
```

Signature of the Faculty

EXPERIMENT NO: 10a Date:

Write a C program for File Locking

Aim:-Write a C program for File locking using semaphore

```
#include<stdio.h>
#include<stdlib.h>
#include<error.h>
#include<sys/types.h>
#include<sys/ipc.h>
#include<sys/sem.h>
int main(void)
{
key_t key;
int semid;
union semun arg;
if((key==ftok("sem demo.c","j"))== -1)
{
perror("ftok");
exit(1);
}
if(semid=semget(key,1,0666|IPC_CREAT))== -1)
{
perror("semget"):
exit(1);
}
arg.val=1;
if(semctl(semid,0,SETVAL,arg)== -1)
{
perror("smctl");
exit(1);
}
return 0;
}
```

Output:

Assignment:

Sno	Task	Date	Sign	Remark
1	Write a program using the simpler semaphore operation			
2	Write a program to create a semaphore			

Signature of the Faculty

EXPERIMENT NO: 10b Date:

Aim:-Write a C program that receives a message from message queue and display them

Algorithm:

Step 1:Start
Step 2:Declare a message queue structure
typedef struct msgbuf {
long mtype;
char mtext[MSGSZ];
} message_buf;
Mtype =0 Retrieve the next message on the queue, regardless of its mtype.
PositiveGet the next message with an mtype equal to the specified
msgtyp.
Negative Retrieve the first message on the queue whose mtype field is
less than or equal to the absolute value of the msgtyp argument.
Usually mtype is set to1
mtext is the data this will be added to the queue.
Step 3:Get the message queue id for the "name" 1234, which was created by the server
key = 1234
Step 4 : if ((msqid = msgget(key, 0666< 0) Then print error
The msgget() function shall return the message queue identifier associated with the argument key.
Step 5: Receive message from message queue by using msgrcv function
int msgrcv(int msqid, void *msgp, size_t msgsz, long msgtyp, int msgflg);
#include < sys/msg.h>
(msgrcv(msqid, &rbuf, MSGSZ, 1, 0)
msqid: message queue id
&sbuf: pointer to user defined structure MSGSZ: message size
Message type: 1
Message flag:The msgflg argument is a bit mask constructed by ORing together zero or
more of the following flags: IPC_NOWAIT or MSG_EXCEPT or MSG_NOERROR
Step 6:if msgrcv <0 return error
Step 7:otherwise print message sent is sbuf.mext
Step 8:stop

Program:

```
//IPC_msgq_send.c

#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#define MAXSIZE     128
void die(char *s)
{
  perror(s);
  exit(1);
}
```

```
typedef struct msgbuf
{
   long    mtype;
   char    mtext[MAXSIZE];
};

main()
{
   int msqid;
   int msgflg = IPC_CREAT | 0666;
   key_t key;
   struct msgbuf sbuf;
   size_t buflen;

   key = 1234;

   if ((msqid = msgget(key, msgflg )) < 0)   //Get
the message queue ID for the given key
     die("msgget");

   //Message Type
   sbuf.mtype = 1;

   printf("Enter a message to add to message
queue : ");
   scanf("%[^\n]",sbuf.mtext);
   getchar();

   buflen = strlen(sbuf.mtext) + 1 ;

   if (msgsnd(msqid, &sbuf, buflen,
IPC_NOWAIT) < 0)
   {
       printf ("%d, %d, %s, %d\n", msqid,
sbuf.mtype, sbuf.mtext, buflen);
       die("msgsnd");
   }

   else
       printf("Message Sent\n");

   exit(0);
}
```

Program:

```
//IPC_msgq_rcv.c

#include <sys/types.h>
#include <sys/ipc.h>
```

```
#include <sys/msg.h>
#include <stdio.h>
#include <stdlib.h>
#define MAXSIZE     128

void die(char *s)
{
  perror(s);
  exit(1);
}

typedef struct msgbuf
{
    long    mtype;
    char    mtext[MAXSIZE];
} ;
main()
{
    int msqid;
    key_t key;
    struct msgbuf rcvbuffer;

    key = 1234;

    if ((msqid = msgget(key, 0666)) < 0)
      die("msgget()");

     //Receive an answer of message type 1.
if (msgrcv(msqid, &rcvbuffer, MAXSIZE, 1, 0) < 0)
      die("msgrcv");

    printf("%s\n", rcvbuffer.mtext);
    exit(0);
}
```

Output:

```
vgupta> ipcs -q

------ Message Queues --------
key        msqid      owner      perms      used-bytes   messages
0x000004d2 9240578    vgupta80   666        12           1
```

This depicts that there is one message in the queue

```
------ Message Queues --------
key        msqid      owner      perms      used-bytes   messages
0x000004d2 9240578    vgupta80   666        0            0
```

The message has been removed

Signature of the Faculty

EXPERIMENT NO: 11 Date:

Aim:-Write a C program that illustrates two processes communicating using Shared memory

Algorithm:-

step1.Start
step 2.Include header files required for the program are
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <unistd.h>
#include <string.h>
#include <errno.h>
step 3.Declare the variable which are required as
pid_t pid
int *shared /* pointer to the shm */
int shmid
step 4.Use shmget function to create shared memory
#include <sys/shm.h>
int shmget(key_t key, size_t size, int shmflg)
The shmget() function shall return the shared memory identifier associated with key The argument key is equal to IPC_PRIVATE. so that the operating system selects the next available key for a newly created shared block of memory. Size represents size of shared memory block Shmflg shared memory permissions which are represented by octal integer
shmid = shmget (IPC_PRIVATE, sizeof(int), IPC_CREAT | 0666);
print the shared memory id
step 5.if fork()==0 Then
begin
shared = shmat(shmid, (void *) 0, 0)
print the shared variable(shared) *shared=2
print *shared sleep(2)
print *shared
end
step 6.else
begin
shared = shmat(shmid, (void *) 0, 0)
print the shared variable(shared)
print *shared sleep(1) *shared=30
printf("Parent value=%d\n", *shared);
sleep(5)
shmctl(shmid, IPC_RMID, 0)
end
step 7.stop.

Sha.c

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <unistd.h>
#include <errno.h>
```

```
int main(void) {
pid_t pid;
int *shared; /* pointer to the shm */ int shmid;
shmid = shmget(IPC_PRIVATE, sizeof(int), IPC_CREAT | 0666); printf("Shared Memory
ID=%u",shmid);
if (fork() == 0) { /* Child */

/* Attach to shared memory and print the pointer */ shared = shmat(shmid, (void *) 0, 0);
printf("Child pointer %u\n", shared); *shared=1;
printf("Child value=%d\n", *shared); sleep(2);
printf("Child value=%d\n", *shared); } else { /* Parent */
/* Attach to shared memory and print the pointer */ shared = shmat(shmid, (void *) 0, 0);
printf("Parent pointer %u\n", shared); printf("Parent value=%d\n", *shared); sleep(1);
*shared=42;
printf("Parent value=%d\n", *shared); sleep(5);
shmctl(shmid, IPC_RMID, 0);
}

}
```

Output:

```
$cc shared_mem.c
$ ./a.out
Shared Memory ID=65537Child pointer 3086680064 Child value=1
Shared Memory ID=65537Parent pointer 3086680064 Parent value=1
Parent value=42 Child value=42
```

Viva questions

1. **define shared memory**
2. what are file locking functions. 3.what are shared memory system calls.
4. define internet domain sockets
5. Difference between internet and unix domain sockets.

Assignment:

Sno	Task	Date	Sign	Remark
1	Write a program to demonstrate communication of two different process via shared memory			
2	Write a program to demonstrate that the shared memory created will be available even after the process which created is exited.			

Signature of the Faculty

EXPERIMENT NO: 12 Date:

Aim:-Write client server programs using c for interaction between server and client process using sockets or Unix Domain sockets

Algorithm:-

Sample UNIX server

```
Step 1:define NAME "socket"
Step 2: sock = socket(AF_UNIX, SOCK_STREAM, 0);
Step 3:if (sock < 0) perror("opening stream socket"); exit(1);
step4: server.sun_family = AF_UNIX;
        strcpy(server.sun_path, NAME);
        if (bind(sock, (struct sockaddr *) &server, sizeof(struct sockaddr_un)))
        {
         perror("binding stream socket");      exit(1);
        }
 step 5: print ("Socket has name %s\n", server.sun_path);
        listen(sock, 5);
step 6: for (;;)
        {
         msgsock = accept(sock, 0, 0);
        if (msgsock == -1)
                perror("accept");
        else
         do { bzero(buf, sizeof(buf));
        if ((rval = read(msgsock, buf, 1024)) < 0)

        perror("reading stream message");
        else if (rval == 0)
        else print ("-->%s\n", buf);
         } while (rval > 0);
        close(msgsock);
        }
close(sock);
unlink(NAME);
 }
```

Step 7:stop

Programs:

Server.c

```
#include <stdio.h>
#include <sys/socket.h>
#include <sys/un.h>
#include <sys/types.h>
#include <unistd.h>
#include <string.h>
```

```
int connection_handler(int connection_fd)
{
 int nbytes;
 char buffer[256];
 nbytes = read(connection_fd, buffer, 256);
 buffer[nbytes] = 0;
 printf("MESSAGE FROM CLIENT: %s\n", buffer);
 nbytes = snprintf(buffer, 256, "hello from the server");
 write(connection_fd, buffer, nbytes);
 close(connection_fd);
 return 0;
}

int main(void)
{
 struct sockaddr_un address;
 int socket_fd, connection_fd;
 socklen_t address_length;
 pid_t child;
 socket_fd = socket(PF_UNIX, SOCK_STREAM, 0);
 if(socket_fd < 0)
 {
  printf("socket() failed\n");
  return 1;
 }

 unlink("./demo_socket");

 /* start with a clean address structure */
 memset(&address, 0, sizeof(struct sockaddr_un));

 address.sun_family = AF_UNIX;
 snprintf(address.sun_path, UNIX_PATH_MAX, "./demo_socket");

 if(bind(socket_fd,
        (struct sockaddr *) &address,
        sizeof(struct sockaddr_un)) != 0)
 {
  printf("bind() failed\n");
  return 1;
 }
```

```
 if(listen(socket_fd, 5) != 0)
 {
  printf("listen() failed\n");
  return 1;
 }

 while((connection_fd = accept(socket_fd,
                        (struct sockaddr *) &address,
&address_length)) > -1)
 {
  child = fork();
  if(child == 0)
  {
   /* now inside newly created connection handling process */
   return connection_handler(connection_fd);
  }
  /* still inside server process */
  close(connection_fd);
 }
 close(socket_fd);
 unlink("./demo_socket");
 return 0;
}
```

Client.c

```
#include <stdio.h>
#include <sys/socket.h>
#include <sys/un.h>
#include <unistd.h>
#include <string.h>
int main(void)
{
 struct sockaddr_un address;
 int socket_fd, nbytes;
 char buffer[256];

 socket_fd = socket(PF_UNIX, SOCK_STREAM,0);
 if(socket_fd < 0)
 {
  printf("socket() failed\n");
  return 1;
 }
```

```
/* start with a clean address structure */
memset(&address, 0, sizeof(struct sockaddr_un));

address.sun_family = AF_UNIX;
snprintf(address.sun_path, UNIX_PATH_MAX, "./demo_socket");

if(connect(socket_fd,
        (struct sockaddr *) &address,
        sizeof(struct sockaddr_un)) != 0)
{
 printf("connect() failed\n");
 return 1;
}
nbytes = snprintf(buffer, 256, "hello from a client");
write(socket_fd, buffer, nbytes);

nbytes = read(socket_fd, buffer, 256);
buffer[nbytes] = 0;

printf("MESSAGE FROM SERVER: %s\n", buffer);

close(socket_fd);
return 0;
}
```

Assignment:

Sno	Task	Date	Sign	Remark
1	Write a program to demonstrate getting and setting the socket options through socket related system call			
2	Write a program to demonstrate bind system call.			

Signature of the Faculty