Week - 01
Overview of Operating System

1. Types of OS Installation

Attended installation

An installation process usually needs a used who attends it to make choices. Such as accepting or declining an end user license agreement specifying performences such as the installation location, supply passwords or assisting in product activation.

Silent installation

* An installation that does not display messages or windows during its program.
* In bigger organizations where thousand of users work, deploying the applications become a typical task and for the reason silent, installation is performed so that the application is installed in background without affecting the work of user.

Unattended installation

* An installation that is performed without user. Interaction doing its programs or with no user present at all.
* One of the reasons to use this approach is to automatic the installation of a large number of system

Expt. No. 01 Reg. No. 2 0 8 I S 2 3 0 1 7

Date / / Page No 01

Headless Installation

* Installation performed without using a computer monitor connected.
* In attended form of headless installation, another machine connects to the target machine and takes over the display output

Schedules or automated Installation

* An Installation process that runs on a process present time or when a predefined condition meet the requirements. as opposed to an Installation process that starts explicity on a user's command.

Clean Installation

A clean installation is one that is done in the absence of any interfering elements such as old version of the computer program being Installation or leftarms from a previous Installation.

Network Installation

Network Installation (net install), in an Installation of a program from a shared network resources that may be done by Installaxing a minimal system before proceeding to download further packages as the network

2. Boot Method

Booting is the process of starting a computer as initiated via hardware such as a button or by a software command.

Types of Booting

→ Warm Booting :- The warm booting is that in which system starts from the starting or from an initial state means.

* In the warm booting the system will be started from it's beggining state means, first, the user will press the power button, then this will read all the instructions from the ROM and the operating system will be automatic-ally get loaded into the system (RAM)

→ Cold Booting :- The cold Booting is that in which system automatically starts when the system is in a running state.

3. File system and Formatting

File System

→ A file system is a process of managing how and where data on a storage disk, which is also refered to as file management or Fs

→ It is a logical disk components that compresses files separated into groups. which is known as direction

→ The file system enables user to view a file in the current directory as files are often managed in a hierarchy.

Expt. No. 01 Reg. No. 208IS23017

Date / / Page No 02

Example of file system

→ FAT :- FAT is a type of file system, which is develo-ped for hard drives. It stands for file allocation table.

FAT8, FAT12, FAT32, & FAT16. are all the different type of FAT

→ GFS :- A GFS is a file system which stands for global file system. It has the ability to make enable multiple computer's to act an integrated machine

→ HFS :- HFS is the file system that is used on a Macin-tosh computer for creating a directory at the time a hard disk is formatted

→ NTFS :- ~~NTFS~~ UDF is the file system stands for universal Disk format and used first developed by OSTA (Optical storage technology association for ensuring consistency among data written to several optical media.

→ NTFS :- NTFS is the file system, which stands for NT file system and stores and retrives files on win-dows NT operating system and other version of windows like windows 2000, windows XP, windows 7, and windows 10.

Formatting

Formatting is a process of preparing the storage device to store the data. Formatting storage device will erase the earlier contexnts of the device.

4. Post Installation tasks :-

Post Installation tasks in the set of steps to be carried out to ensure that the Installation in complete and went smoothly.

· Post Installation Tasks for ubuntu operating system

-> online accounts

The first step allows user to configure. online, account in case user wants to integerate the desktop with different services.

-> Livepatch

Livepatch in a service that allows the Installation of some updates that could generally require a system reboot of such as those of the kernal.

-> Help improve ubuntu

In this step unix can choose wheather or not to send data from his system to ubuntu.

Expt. No. 01 Reg. No. 2 0 8 I S 2 3 0 1 7

Date / / Page No 03

Privacy :

If requires, a user can es enable location services so that apps on can determine user geographic location. dll the applications installed by default on ubuntu are free software.

You are ready to start

The last screen shows some featured application - some of which are not free software with the option to open the software to center to install them.

Introduction to Unix

* UNIX is an operating system
* An operating system is a set of programs that act as a link between computer and the user
* The operating system (OS) manages the resources of a computer
* Examples of computer resources are :- CPU, RAM, disk memory, printers, display, key board etc.

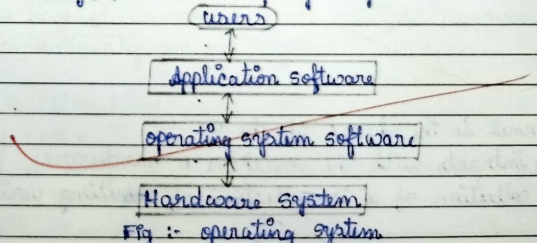
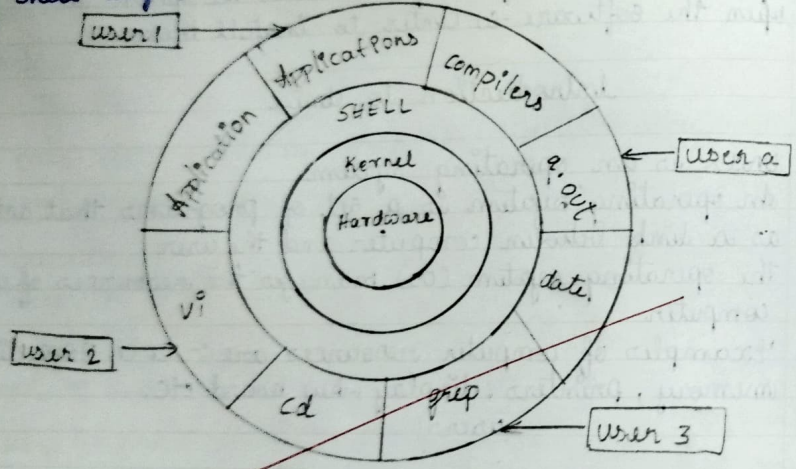


Fig :- operating system

* UNIX OS allows complex tasks to be performed with a few key strokes
* UNIX OS doesn't tell or warn the user about the consequences of the command.
* There are various UNIX variants available in the market. Solaris UNIX, AIX, HP UNIX and BSD are few examples. Linux is also a flavour of unix which is freely
* The UNIX operating system consist of a kernel layer, a shell layer, an application layer and files



Kernel
* The kernel is the heart of the OS
* It ~~at~~ interacts with the machine's hardware
* It is collection ~~of~~ of its collection of routing written in C

Expt. No. 01 Reg. No. 208I52301I7
Date / / Page No 04

* It is loaded into memory when the system is booted

SHELL
* The shell interacts with the user
* The shell is a command line interpreter.
* There are multiple shells that are used by the UNIX
  Example :- Bourne shell, C shell, Korn shell,
  Bourne Again Shell.

File :
* A file is an array of bytes that stores information
* All the data of unix is organized into files
* All files are then organized into directories

Features of UNIX

* Multi user
  * UNIX is a multiprogramming system
  * Multiple users can access the system by connecting to points known as terminals
  * Several users can even multiple programs simultaneously on one system

* Multitasking
  * UNIX is a multitasking system
  * A single user can also run multiple task. can currently
  For example :-

* edit a file
* print another file one on the printer
* send email to a friend.
* browse www.

In a multitasking environment a user sees one job running in the foreground, the rest run in the background users can switch jobs between background and foreground, suspend or even terminate them

* Pattern Matching :-
* UNIX has very sophisticated pattern matching features
* Regular expressions are a features of UNIX
* Regular Expressions describe a pattern to match, a sequence of characters, not words, within a line of text.

* Portable :-
* UNIX can be installed on many hardware platforms
* UNIX operating system is written in C. language, hence it is more portable than the Os.

* UNIX Toolkit :-
Unix offers facility to add and remove many applications as when required tools include
* General purpose tools
* Text manipulation tools
* Compilers / interpreter
* Networked application
* System administration tools

* Programming facility :-
* The unix shell is also a programming language; it was designed for programmer not for end user.
* It has all the necessary ingredients. like control structure loop and variables that establish powerful programming language
* These features are used to design shell scripts program that can also Provoke unix commands
* Many of the system's functions can be controlled and automated by using these all scripts

* Documentation.
* The principal on-line help facility available is the man command, which remains the most important references for commands and their configuration files.
* Apart from the man documentation, there's a vast ocean of unix resources available on the internet.

Expt. No. 01 Reg. No. 2 0 8 I S 2 3 0 1 7
Date / / Page No. 05

Expt. No. 01 Reg. No. 2 0 1 5 2 3 0 1 1

Date / / Page No 06

* Posix and single unix specification
  * The portable operating system interface is a family of standards specified by IEEE for maintaining compatibility between operating system.
  -> Two of the most important standards from POSIX are
  POSIX.1 :- Specifies the C applications program interface
  POSIX.2 :- Deals with the shell and utilities
  -> In 2001 joint initiative of X/ open and IEEE resulted in the unification of two standards
  -> This is a Single UNIX Specifications version 3.

Seen

Week - 02

Virtualization

Install and configure virtual ~~machine~~ - Virtual box

-> A terminal emulator is a computer program that reproduces a video terminal within some other display structure

-> The terminal emulator allows an end-user to access the console all will as its applications such as text, user interface and command-line interface

Example for terminal Emulator :-
Terminator, RoxTurm, Eform, Tilik, LxTerminde, Konnde, Kitty, St, Gname-Terminal, Terminology, Guake, Terminal, XTerm, Tilyterm, Extraterm, mate-terminal, DDM Term, Termkit.

Example 1 :- XTerm

-> The xterm terminal application is a standard terminal emulator for the X window system

-> Installation :-
Sudo apt :- get update
Sudo apt :- get install xterm

Example 2 :- mate-terminal

-> mate terminal is a terminal emulation application to a unix shell in the MATE environment

Expt. No. 02 Reg. No. 2 0 8 I 5 2 3 0 1 7
Date / / Page No 07

-> mate terminal also has the ability to use multiple terminal in a single window and supports management of different configuration (profile)
-> mate terminal is a fork of Gname - Terminal
-> Installation
Sudo apt - get update -4
Sudo apt - get install mate - terminal

Significance of Man Command
-> Man command in linux is used to display the user manual of any command that runs on the Terminal.
-> It provides a detailed view of the command which includes NAME, SYNOPSES, DESCRIPTION, OPTION, EXIT, STATUS, RETURN VALUES, ERRORS, FILES, VERSIONS, EXAMPLE, AUTHORS, and SEE also.

-> Every manual is divided into the following sections:-
* Executable programs or shell commands
* System calls [Functions provided by the kernal]
* Library calls [Functions within program libraries]
* Games
* Special files
* File formats and conventions &etc / password
* Miscellaneous
* System administration commands

Syntax
Man [SECTION-NUM] [COMMAND NAME]

Week - 03

File System

File and directory commands

ls (list)

> Use ls without any arguments to display current directory contents.

> ls with the -a option. files all begin with a "dot", which indicates they are "hidden" files.

→ ls -a

> ls with -F. this command is useful for distinguishing between directories, ordinary files, and executable files

→ ls -F

> ls with -l to obtain a "long" listing of files. An explanation of the information it provides appears below. ls -l

```
-rwxr-xr-x  1 jsmith  staff   43  mar 23  18:14  prog1
-rw-r--r--  1 jsmith  staff  10030 mar 22  20:41  sample.f
drwxr-sr-x  2 jsmith  staff   512  mar 23  18:07  subdir1
drwxr-sr-x  2 jsmith  staff   512  mar 23  18:06  subdir2
drwxr-sr-x  2 jsmith  staff   512  mar 23  18:06  subdir3
    1       2  3      4       5      6        7
```

1 = access mode / permission
2 = Number of links
3 = owner.

Expt. No. 03 Reg. No. 208I52301I7
Date / / Page No 08

4 = group
5 = Size (in bytes)
6 = date & time of last modification
7 = name of file

> ls with R. Recursive listings which will display files and folders inside the folders recursively
→ ls -R
→ ls -R1

Pwd (Present working directory)

→ will give the present working directory path information

MKdir (Make directory)

→ MKdir will create a new directory (folder)
→ Syntax :- MKdir directory name (folder name)
Ex :- MKdir ~~dir~~ poorvika

Cd (Change directory)

→ Cd is used to change directory
→ Change home directory
> cd
> cd ~
→ Change to a subdirectory within user home directory
> cd poorvika / computer
> pwd

→ Go up on level back to the current directories parent directory
> cd..
> pwd

→ Change to the root (top-most) directory
> cd /
> pwd

rmdir (Remove directory)
→ will remove (delete) the directory
→ make sure that folder is empty before issuing rmdir command.
→ Make sure current directory is not part of the directory being deleted
> rmdir poorvika
> rmdir /poorvika/computer.

File manipulation commands :-
Creating File
> touch :- will create empty files
Ex :- touch a.txt b.txt
> Creating file and add contents at the end press Enter key and ctrl+z
Ex :- Cat > a.txt
Welcome to operating system lab
Enjoy Typing commands
ctrl+z

Expt. No. 03 Reg. No. 208IS23017
Date / / Page No 09

Cat a.txt

Display the contents of files :
Syntax :- Cat filename
Cat /Path/file name
Ex :- > Cat a.txt
> Cat Polytechnic/computer/a.txt.

rm (remove file)
> will remove (delete) the files
> Syntax : rm filename
rm path/filename
Ex :- rm a.txt

CP (copy)
> Used to create a copy of a existing file
> Copy an existing file in current directory to another file in the current directory. Make sure that the file to be copied must be exists in specified location
> Syntax : CP file1 file2
CP -i file1 file2
CP -R subdir1 subdir4
CP polytechnic/civil/a.txt.
> Change to another directory
>cd ~/polytechnic
>pwd
> Change to another one of subdirectories
>cd ~/polytechnic/computer

> pwd

mv (move)
→ used to remove the file
→ It can be used to rename file. Directory (folder) also.
ex:- mv old filename new filename
ls
ex:- mv oldfoldername new foldername.
ls

more
→ more in a filter for paging through text one screenful at a time
→ use the more command to read to file
→ more filename

less
→ less is the opposites of more command
→ Use the more command to read a file
→ less command.

cmp (compare)
→ Compare two files byte by byte
→ Syntax: cmp - b file 1 file 2
$ cmp file 1 file 2 ↵
file 1 file 2 differ : byte 7, line 2

head and tail
→ output the first last part of files (by default first / last 10 lines)
> head filename
> tail filename

Expt. No. 03 Reg. No. 2 0 8 I S 2 3 0 1 7
Date / / Page No 10

ex:- head -5 a.txt will display first 5 lines of a.txt
tail -5.txt will display last 5 lines of a.txt

1) $ head file 3
animals list
dog
goat
lion
tiger
-11-

2) $ cat > file 3
animals list
dog
goat
lion
tiger
-11-

3) $ head -4 file 3
animals list
dog
goat
lion
-11-

tail :- It is used to display the last 7 few lines of a file. By default it display the last 10 lines of a file
Syntax: tail <filename>
$ tail file 3
goat
lion
tiger
Cat
elephant
Cheetha
rabbit
-11-

we can also specify the number of lines we want to display using '-n' option

Syntax : tail -n <filename>

```
$ tail -5 file3
  cheetha
  rabbit
  deer
  sheep
  zeebra
```

File Permission

> There are three users in linux namely owner (o), group (g), other (o)

> For all user can be indentified with letter a

> Number assigned for permissions are read (4), write (2), execute (1)

> All three users will have three possible permission namely read (r), write (w), execute (p)

> Command used to change permission of file is chmod

Syntax :-

chmod filepermissionpattern filename

operation -> assign (+), deny (-)

Relative permission

1) To give execution permission for owner

2) To give writing permission for group.

```
$ ls -l
total 12
-rw-rw-r--1 aishu aishu 22 Apr 18 22:12 file1
-rw-rw-r--1 aishu aishu 24 Apr 18 22:13 file2
-rw-rw-r--1 aishu aishu 78 Apr 18 22:17 file3
$ chmod u+x file1
$ ls -l
total 12
-rwxrw-r--1 aishu aishu 22 Apr 18 22:12 file1
-rwxrw-r--1 aishu aishu 24 Apr 18 22:13 file2
-rwxrw-r--1 aishu aishu 78 Apr 18 22:17 file3
```

Absolute permission

1) To give all permission to all

2) To remove writing permission to group

Binary	Octal	Permission
000	0	---
001	1	--x
010	2	-w-
011	3	-wx
100	4	r--
101	5	r-x
110	6	rw-
111	7	rwx

File compression

gzip : it compress the files

Syntax : gzip <filename>

Option

Expt. No. 03 Reg. No. 2 0 8 I 6 2 3 0 1 7

Date / / Page No. 11

-l :- to see the compressed and uncompressed file size and ratio

-d - to decompress

$ ls -l

total 16

-rw-rw-- 1 aishu aishu 486 Apr 19 00:12 database

-rwxr-xr-- 1 aishu aishu 22 Apr 18 22:12 file1

$ gzip database

ls -l

-rw-rw-r-- 1 aishu aishu 26: Apr 19 00:12 file1

-rwxr-xr-- 1 aishu aishu 22 Apr 18 22:12 file1

tar :- for creating a disk archieve

option

-c :- creation archieve eg :- tar -cvf

-x :- Extract file from archieve eg :- tar -xvf

-t :- display files an archieve eg :- tar -tvf

$ $ tar -cvf file1. tar file2 file3

file2

file3

$ tar -xvf file4. $ tar

file1

file2

Expt. No. 03 Reg. No. 2 0 2 1 5 2 3 5 1 7

Date / / Page No. 12

Text processing commands

Sort : It is used to sort a file arranging the records in a particular order

Syntax :- sor [options] filename

options

1) -o :- used to save sorted content in specified file

Syntax :- Sort -o database database

2) -r :- used to sort the file contents in descending order

eg :- sort -r database

3) -n :- If the file contains numbers based on number values file contents can be sorted

eg :- sort -n database

4) -k :- Sort the files contents based on kth field values

eg :- sort -k 2 database

5) -u :- will remove duplicate entries in file while sorting

eg :- sort -u database

$ sort -n database

ID	Name	Designation	department	salary
101	aishu	Engineering	IS	56000
102	ammu	Developer	IT	30000
103	anulya	HR	HR	35000

Uniq :- used to remove the duplicate lines from a sorted file
Syntax :- using -c filename

Join :- Join allows merging two files in a meaningful fashion

$ Cat file 1	$ Cat file 2	$ cat file 3
hello	hello	hello
mysore	Karnataka	welcome
Welcome	Welcome	

Grep :- A multi purpose file search tool that uses regular Expression
Syntax :- $ grep opt. pattern filename

options
-i = (ignore case)
-v = inverse
-n = line numbers
-c = count
-l = list
-e = multiple (To select more than one lines)
-f = file

$ grep "Sales" database
109 | arun | sales head | sales | 450
grep -i "manager" database
109 | John | Manager | Sales | 65000

Expt. No. 03 Reg. No. 2 0 8 I S 2 3 0 1 7
Date / / Page No. 13

option -n : line number

tr
Character translation filter
Syntax :- tr pattern filename
$ tr '[:]' '[;]' < file 2
hello
Karnataka
$ tr '[a-z]' '[A-Z]' < file 2
HELLO
KARNATAKA

Comm
It requires 2 sorted files and list the different entries in different column
① lines unique to file 1 ② lines unique to file 2
③ lines common on both
$ Comm file 1 file 2
hello
Karnataka
Mysore
Welcome

→ diff :- It shows the difference b/w the 2 files

```
$ diff file1 file2
2, 3d2
< mysore
< welcome
---
> Karnataka
> welcome
```

WC :- We gives a "word count on a file".

Syntax :- WC <filename>

options :

WC - L : gives only the line count

WC - W : gives only the word count

WC - C : gives only byte count

```
$ WC file1
  3 3 22 file1
$ WC -l file1
  3 file1
$ WC -w file1
  3 file1
$ WC -c file1
  22 file1
```

Cut :- A tool for extracting field from files

Syntax : $ cut [optional] filenames)

Expt. No. 03 Reg. No. 2 0 8 I S 2 3 0 1 7

Date / / Page No. 14

options

-c :- It is used to cut a file column wise specifying column numbers

-f :- along with the delimeter [vertical bar] options -d used to cut file fieldwise by specifying the field nos.

Paste

Paste what we had cut [will display 2 files side by side by pasting.

Syntax :- $ Paste filename1 filename2

```
$ paste db db1
name        department
john        Sales
aritma      Engineering
amrutha     MBBS
Gagana      NCC
Anaiya      NCC
```

Pr - paginating files

Prepare a file for printing by adding suitable head features and formatted text syntax - $pr filename(s)

→ Top - bottom - adds 5 lines (time, date, filename page no)

options

-k → (where k is an integer) print the output in k Column

-d → Display the output in double line

-n → display the output in line members

-on → (where -n is an integer) after line by -n space increasing the left margin of page.

$pr database.

ID	Name	Department	Salary
101	Aishu	Engineering	20000
102	Shathu	Designer	30000

option -k

$ pr -k database

ID	Name	102	asmitha	104
101	John	103	michal	105

-d

$ pr -d database

ID	Name	Department	Salary
101	Aishu	IT	Engineering
102	Ammu	IT	Engineering

Expt. No. 03 Page No. 2 0 2 1 S 2 3 0 1 7

Date / / Page No. 15

-n

$ pr -n database

1	ID	HR	Department	Salary
2	101	Analyst	Market	1000
3	102	Manager	Sales	2000

-on

$ pr -o10 database

ID N 101 i 102 a 103 m 104.

```
buntu@ubuntu-virtualbox:~/pooja$ ps
  PID TTY          TIME CMD
 2651 pts/0    00:00:00 bash
 2695 pts/0    00:00:00 ps
```

Expt. No. 04 Reg. No. 2 0 8 I 5 2 3 0 1 7

Date / / Page No 16

Week - 04
Process Management

Process

> A process is any active (running) instance of a program. In other words process is a program in execution

> A new process can be created by the fork() system call

> The new process consists of a copy of the address space of the original process fork().

> Each process is given a unique process identification (PID)

> PID is usually five-digit number

- This number is used to manage each process
- User can also use the process name to manage process

> **PS command**

This command is used to display attributes of process that are running currently.

PID -> Process Identification number
TTY -> Terminal
Time -> CPU Time taken by the process
CMD -> Process name

```
$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
ubuntu      2651    2556  0 10:52 pts/0    00:00:00 bash
ubuntu      2696    2651  0 10:53 pts/0    00:00:00 ps -f
```

```
ubuntu@ubuntu-VirtualBox:~/poobin$ ps -a
    PID TTY          TIME CMD
   1325 tty2     00:00:00 gnome-session-b
   2697 pts/0    00:00:00 ps
```

```
ubuntu@ubuntu-VirtualBox:~/poobin$ ps -ax
    PID TTY      STAT   TIME COMMAND
      1 ?        Ss     0:00 /sbin/init splash
      2 ?        S      0:00 [kthreadd]
      3 ?        I<     0:00 [rcu_gp]
      4 ?        I<     0:00 [rcu_par_gp]
      5 ?        I<     0:00 [slub_flushwq]
      6 ?        I<     0:00 [netns]
```

```
$ ps -aux
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.3 166660 11640 ?        Ss   10:29   0:00 /sbin/init splash
root           2  0.0  0.0      0     0 ?        S    10:29   0:00 [kthreadd]
root           3  0.0  0.0      0     0 ?        I<   10:29   0:00 [rcu_gp]
```

```
ubuntu@ubuntu-VirtualBox:~/poobin$ ps -ef
UID          PID    PPID  C STIME TTY          TIME CMD
root           1       0  0 10:29 ?        00:00:00 /sbin/init splash
root           2       0  0 10:29 ?        00:00:00 [kthreadd]
root           3       2  0 10:29 ?        00:00:00 [rcu_gp]
root           4       2  0 10:29 ?        00:00:00 [rcu_par_gp]
```

```
ubuntu@ubuntu-VirtualBox:~/poobin$ ps -ux
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START
ubuntu      1256  0.0  0.3  17956 10696 ?        Ss   10:29
ubuntu      1257  0.0  0.2 169952  5964 ?        S    10:29
ubuntu      1263  0.0  0.2  39568  6272 ?        S<sl 10:29
```

Expt. No. 04 Reg. No. 208152301017

Date / / Page No 17

2) ps -f

ps -f (full) option can be used to get a detailed list of attributes of all process including parent process.

3) ps -A

ps -A command is used to display information about all running process on the system

4) ps -ax

ps -ax provides a comprehensive list of all running process on the system, regardless of terminal or user

5) ps - aux

used to detailed information of all process running on system

6) ps -ef

7) ps - ux

ps -ef and ps - ux :- Both gives detailed information of all running process

```
ubuntu@ubuntu-VirtualBox:~/poobin$ touch poobin
ubuntu@ubuntu-VirtualBox:~/poobin$ cat>poobin
blahhh blahhh blahhh
blah blah blahhhh
blah blah blahhhhh
^Z                              cat > poobin
[1]+  Stopped
ubuntu@ubuntu-VirtualBox:~/poobin$ cat poobin
blahhh blahhh blahhh
blah blah blahhhh
blah blah blahhhhh
ubuntu@ubuntu-VirtualBox:~/poobin$ jobs
                                cat > poobin
[1]+  Stopped
ubuntu@ubuntu-VirtualBox:~/poobin$ bg
[1]+ cat > poobin &
ubuntu@ubuntu-VirtualBox:~/poobin$ fg
cat > poobin
^Z
[1]+  Stopped                   cat > poobin
```

```
ubuntu@ubuntu-VirtualBox:~$ top
top - 18:57:01 up 27 min,  1 user,  load average: 0.02, 0.04, 0.03
Tasks: 191 total,   2 running, 188 sleeping,   1 stopped,   0 zombie
%Cpu(s):  0.5 us,  0.0 sy,  0.0 ni, 99.4 id,  0.0 wa,  0.0 hi,  0.1 si,  0.0 st
MiB Mem :   2847.7 total,   1229.3 free,    733.2 used,    885.2 buff/cache
MiB Swap:   1162.4 total,   1162.4 free,      0.0 used.   1899.0 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
   1439 ubuntu    20   0 4862876 365956 133276 S   3.6  12.5   1:00.94 gnome-shell
    469 systemd+  20   0   14836   6784   6016 S   0.3   0.2   0:01.92 systemd-oomd
   2556 ubuntu    20   0  629380  53912  41108 S   0.3   1.8   0:01.11 gnome-terminal-
      1 root      20   0  166660  11640   8184 S   0.0   0.4   0:00.89 systemd
      2 root      20   0       0      0      0 S   0.0   0.0   0:00.00 kthreadd
      3 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_gp
      4 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_par_gp
      5 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 slub_flushwq
      6 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 netns
      8 root      20   0       0      0      0 I   0.0   0.0   0:00.94 kworker/u6:0-events_power_efficient
     10 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 mm_percpu_wq
     12 root      20   0       0      0      0 I   0.0   0.0   0:00.00 rcu_tasks_kthread
     13 root      20   0       0      0      0 I   0.0   0.0   0:00.00 rcu_tasks_rude_kthread
     14 root      20   0       0      0      0 I   0.0   0.0   0:00.00 rcu_tasks_trace_kthread
     15 root      20   0       0      0      0 S   0.0   0.0   0:00.07 ksoftirqd/0
     16 root      20   0       0      0      0 I   0.0   0.0   0:00.91 rcu_preempt
     17 root      rt   0       0      0      0 S   0.0   0.0   0:00.05 migration/0
     18 root     -51   0       0      0      0 S   0.0   0.0   0:00.00 idle_inject/0
     19 root      20   0       0      0      0 S   0.0   0.0   0:00.00 cpuhp/0
     20 root      20   0       0      0      0 S   0.0   0.0   0:00.00 cpuhp/1
     21 root     -51   0       0      0      0 S   0.0   0.0   0:00.00 idle_inject/1
     22 root      rt   0       0      0      0 S   0.0   0.0   0:00.29 migration/1
     23 root      20   0       0      0      0 S   0.0   0.0   0:00.09 ksoftirqd/1
```

Expt. No. 04 Reg. No. -2 0 8 1 5 2 3 0 1 7

Date / / Page No 18

Jobs fg (foreground), bg (background)

Jobs :- Display current jobs running

Sleep :- sleep is a command which is used to create dum-my b jobs which runs for 500 runs → create job using sleep and get jobid as [2] [here]

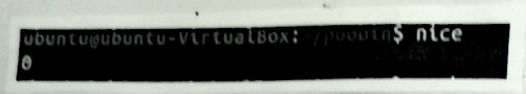
bg [background]:

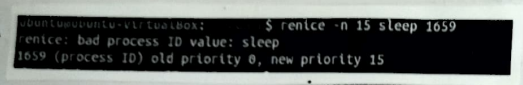
* bg is used to place foreground jobs in background
* fg be syntax :- bg [job id]
* here by 2 or bg %2 can be used to put it in background
* Fg is command that moves background process on current linux shell to the foreground

  Syntax :- fg - job [id]

* here sleep 500 - job id - 2
  - 7 fg 7.2 or fg 2

Top :- Top command is linux is popular ubility used to monitor system process in real-time, when you run the top command it displays a dynamic interactive viewer of the currently running process on your system





Expt. No. 04 Reg. No. 2 0 8 I S 2 3 0 1 7

Date / / Page No 19

Nice and Renice :-

→ Nice : Nice is a value associated with a process that determines it's priority for CPU scheduling

→ The 'Nice' command is used to launch a process with a specified nice value.

Renice

→ Renice is a command used to change the nice value of an already running process it allows you to adjust the priority of a process while it's running.

→ renice -n 5 process name or job name

→ here is the above example increasing priority of process with pid

you can also used command

renice -n -5p

Kill command

Kill command used to terminate or send signals to process it allows you to manage running process effectively

- Kill Pid
  → Sends the default signal to the process with specified pid (process id) causing it to terminate

Expt. No. 04 Reg. No. 2 0 3 1 5 2 3 0 1 7

Date / / Page No 20

2) Kill -9 pid :-

Kill -9 pid is used to forcefully terminated a process with specified process id

Seen

Week - 05

Introduction to Process Synchronization

Commands to exhibit thread concepts

* Threads are a popular programming abstracting for parallel executing on modern operating systems.
* In Linux threads (also called lightweight processes (LWP) created within a program will have the same "thread group ID" as the program's PID

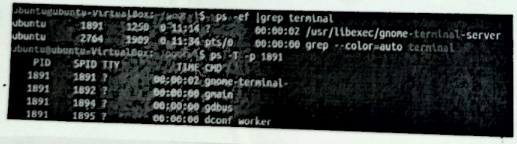
Using the PS

The "-T" option for the PS command enables Thread views

PS -T -P <pid>

* First find the all process containing word terminal

PS -ef | grep terminal



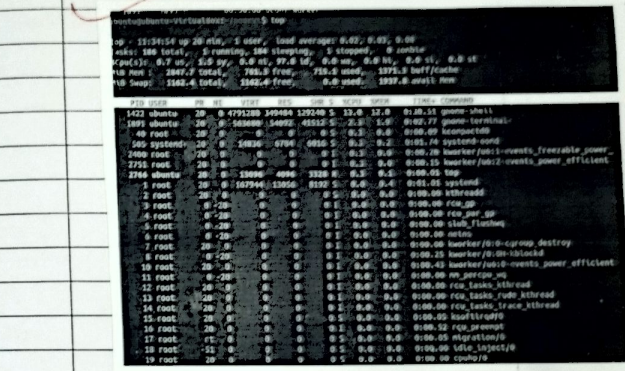
Expt. No. 05 Reg. No. 2 0 8 I S 2 3 0 1 7

Date / / Page No 21

using the top

The top command can show a real time view of individual threads

Example 1 : top



Example 2 : top -h -p 1422

Seen

Week -06

Memory Management.

1) Free command

Syntax :- free -m

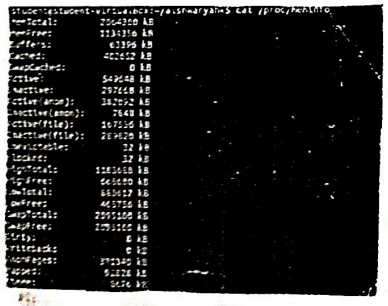
* The m option displays all data in MBs.

```
student@student-VirtualBox:~/aishwaryahk$ free -m
              total        used        free      shared  buffers     cached
Mem:          2015         909        1106           8          61         393
-/+ buffers/cache:          454        1561
Swap:         2045           0        2045
student@student-VirtualBox:~/aishwaryahk$
```

2) /Proc/meminfo

* check memory usage by reading the file at /proc/meminfo

Syntax:- /Proc/meminfo



Expt. No. 6 Reg. No. 2 0 8 C S 2 2 0 0 2

Date 27/2/24 Page No 18

3) vmstat

- vmstat -s
- The top few lines indicate total memory, free memory etc and so on.

```
student@student-VirtualBox:~/aishwaryahk$ vmstat -s
      2064300 K total memory
       931840 K used memory
       551136 K active memory
       297668 K inactive memory
      1133260 K free memory
        63396 K buffer memory
       402652 K swap cache
      2095100 K total swap
            0 K used swap
      2095100 K free swap
        15050 non-nice user cpu ticks
          112 nice user cpu ticks
         4871 system cpu ticks
      3173345 idle cpu ticks
         1029 IO-wait cpu ticks
            0 IRQ cpu ticks
            9 softirq cpu ticks
            0 stolen cpu ticks
```

4) RAM Information:-

- To find out hardware information about the installed RAM, use the demicode command

Syntax:- sudo dmidecode -t 17

```
student@student-VirtualBox:~/aishwaryahk$ sudo dmidecode -t 17
[sudo] password for student:
# dmidecode 2.12
SMBIOS 2.5 present.

student@student-VirtualBox:~/aishwaryahk$
```

Week - 07

Shell Script

Write a shell program to perform arithemetic operations

```
read -p 'Enter a:' a
read -p 'Enter b:' b
add = $((a+b))
echo "Addition of $a and $b is $add"
sub = $((a-b))
echo " Subtraction of $a and $b is $sub "
mul = $((a*b))
echo "multiplication of $a and $b is $mul"
div = $((a/b))
echo " division of $a and $b is $div"
mod = $((a%b))
echo " modular of $a and $b is $mod"
```

Output :

```
$ bash arith
Enter a: 2
Enter a: 3
Addition of 2 and 3 is 5
Subtraction of 2 and 3 is 1
Multiplication of 2 and 3 is 6
Division of 2 and 3 is 0
Modules of 2 and 3 is 2
```

Expt. No. Reg. No.

Date / / Page No

2. Write a shell program to perform relational operator

```
read -p "Enter a:" a
read -p 'Enter b:' b
if (( $a == $b)) ; then
echo "$a is equal to $b
else
echo "$a is not equal to $b"
fi
if (( $a != $b)) ; then
echo "$a is not equal to $b"
else
echo "$a is equal to $b"
fi
if (( $a <= $b)) ; then
echo "$a is less than $b"
else
echo "$a is not less than $b"
fi
if (( $a <= $b)) ; then
echo "$ a is less then or equal to $b"
else
echo "$a is not less than or equal to $b"
fi
if (( $a > $b)) ; then
echo "$ a is greater than $b"
else
echo "$ a is not greater than $b"
```

fi
if (($a >= $b)); then
echo "$a is greater than or equal to $b"
else
echo "$a is not greater than or equal to $b"
fi

Output
$ bash relational
Enter a: 5
Enter b: 9
5 is not equal to 9
5 is not equal to 9
5 is less than 9
5 is less than or equal to 9
5 is not greater than 9
5 is not greater than or equal to 9

3) Write a shell program to perform logical operations

```
read -p 'Enter a:' a
read -p 'Enter b:' b
if (( $a == "true" && $b == "true" )); then
echo Both are true
else
echo Both are not true
fi
if (( $a == "true" || $b == "true" )); then
echo at least are of them is true
else
```

Expt. No. | Reg. No.
Date / / | Page No

```
echo None of them is true
fi
if [[ ! $a == "true" ]]; then
echo "a" was initially false
else
echo "a" was initially true
fi
```

Output

```
$ bash logic
Enter a: true
Enter b: false
Both are true
Atleast one of them is true
a was initially true
```

4. Looping statements in shell scripts

1) While statement

```
a=0
while [ $a -lt 10 ]
do
echo $a
a=$(expr $a + 1)
done
```

output :-

```
$ bash while
 0
 1
 2
 3
 4
 5
 6
 7
 8
 9
```

For statement

```
for a in 1 2 3 4 5 6 7 8 9 10
do
if (( $a == 5 ))
  then
  break
  fi
echo "Iteration no $a"
done.
```

output

```
Iteration no 1
Iteration no 2
Iteration no 3
Iteration no 4
```

Expt. No. | Reg. No.

Date / / | Page No

Implementing for loop with continue statement

```
for a in 1 2 3 4 5 6 7 8 9 10
do
if ((C $a = = 5))
then
  continue
fi
echo "Iteration no $a"
done
```

Output

Iteration no 1
Iteration no 2
Iteration no 3
Iteration no 4
Iteration no 5
Iteration no 6
Iteration no 7
Iteration no 8
Iteration no 9
Iteration no 10

3) Until loop

```
a=0
until [ $a -gt 10]
do
  echo $a
  ((a++))
done
```

output:

0
1
2
3
4
5
6
7
8
9
10