LINUX AND WIN32 PROGRAMMING

PRACTICAL
FILE IT – 361

BACHELOR OF TECHNOLOGY

INFORMATION TECHNOLOGY

UNIVERSITY SCHOOL OF INFORMATION, COMMUNICATION & TECHNOLOGY, DWARKA

GURU GOBIND SINGH INDRAPRASTHA UNIVERSITY

SUBMITTED TO: `

Ms. Shweta Sharma
Assistant Professor
(USICT,GGSIPU)

SUBMITTED BY:

Apoorv Aron

01316401520

B.Tech IT 5th SEM

INDEX

3.	File commands: ● Cat ● cp ● copy ● rm ● wc ● cmp ● diff ● comm	
4.	List Commands: ● ls ● ls-a ● ls -l ● ls -r	
5.	Permissions: ● Chmod ● chgrp ● chown	
6.	Process Commands: ● ps ● top	

	• bg • fg • nice • renice • kill	
7.	Grep Commands:	
8.	Word Count	
9.	Wild characters: *, ?, [ ]	
10.	Introduction to Vi editor and its commands	
11.	WSS to enter two strings	
12.	Write a shell script add, subtract, multiply and divide two numbers.	
13.	Write a shell script to find largest of two numbers.	
14.	Write a shell script to Check whether given number is even or odd.	
15.	Write a Shell script to list all of the directory files in a directory	
16.	Write a shell script to find factorial of a number.	
17.	Write a C Program to design a calculator.	

18.	Write a C program to create a child process and allow the parent to display “parent” and the child to display “child” on the screen.?	

Lab 1

Basic Commands

- **echo**
  - echo command in Linux is used to display line of text/string that are passed as an argument. This is a built-in command that is mostly used in shell scripts and batch files to output status text to the screen or a file.
  - Syntax:

    echo [option] [string]

    echo [string]
  - Options:
    - \b backspace
    - \c produces no further output
    - \e escape
    - \f form feed
    - \n new line
    - \r carriage return
    - \t horizontal tab
    - \v vertical tab
    - -n do not output the trailing newline-e enable interpretation of backslash escapes
    - -E disable interpretation of backslash escapes (default)
    - --help display this help and exit
    - --version output version information and exit
    - If -e is in effect, the following sequences are recognized:

```
> echo "This is Apoorv Aron"
This is Apoorv Aron
~ ----------------------------------------
>
```

- **printf**
  - o "printf" command in Linux is used to display the given string, number or any other format specifier on the terminal window. It works the same way as "printf" works in programming languages like C.
  - o Syntax:

    $printf [-v var] format [arguments]

  - o OPTIONS:
    - --help display this help and exit
    - --version output version information and exit

```
> printf "%s\n" "Stadium"
Stadium
~ ----------------------------------------
>
```

- **bc**

  - bc command is used for command line calculator. It is similar to basic calculator by using which we can do basic mathematical calculations.

  - Arithmetic operations are the most basic in any kind of programming language. Linux or Unix operating systems provides the bc command and expr command for doing arithmetic calculations. You can use these commands in bash or shell script also for evaluating arithmetic expressions.

  - Syntax:

    bc [-hlwsqv] [long-options] [file ...]

  - Options:

    - -h, {- -help } : Print the usage and exit
    - -i, {- -interactive } : Force interactive mode
    - -l, {- -mathlib } : Define the standard math library
    - -w, {- -warn } : Give warnings for extensions to POSIX bc
    - -s, {- -standard } : Process exactly the POSIX bc language
    - -q, {- -quiet } : Do not print the normal GNU bc welcome
    - -v, {- -version } : Print the version number and copyright and quit.

```
> echo "6+21" | bc
27
~ ----------------------------------------
>
```

- **whoami**

  - o whoami command is used both in Unix Operating System and as well as in Windows Operating System.
  - o It is basically the concatenation of the strings “who”,”am”,”i” as whoami.
  - o It displays the username of the current user when this command is invoked.
  - o It is similar as running the id command with the options -un.
  - o The earliest versions were created in 2.9 BSD as a convenience form for who am i, the Berkeley Unix who command’s way of printing just the logged in user’s identity. The GNU version was written by Richard Mlynarik and is part of the GNU Core Utilities (coreutils).
  - o Syntax:

    Whoami

```
> whoami
apoorv
~ ----------------------------------------
>
```

- **clear**

  - clear is a standard Unix computer operating system command that is used to clear the terminal screen. This command first looks for a terminal type in the environment and after that, it figures out the terminfo database for how to clear the screen. And this command will ignore any command-line parameters that may be present. Also, the clear command doesn't take any argument and it is almost similar to cls command on a number of other Operating Systems.

  - Syntax:

    $clear

```
> echo "This is Apoorv Aron"
This is Apoorv Aron
> printf "%s\n" "Stadium"
Stadium
> echo "6+21" | bc
27
> whoami
apoorv
~ ----------------------------------------
> clear
```

```
~ ----------------------------------------
>
```

- **passwd**
    - passwd command in Linux is used to change the user account passwords. The root user reserves the privilege to change the password for any user on the system, while a normal user can only change the account password for his or her own account.
    - Syntax:

      passwd [options] [username]
    - OPTIONS:
        - -k: The option -k, is used to indicate that the update should only be for expired authentication tokens (passwords); the user wishes to keep their non-expired tokens as before.
        - -l: This option is used to lock the specified account and it is available to root only. The locking is performed by rendering the encrypted password into an invalid string (by prefixing the encrypted string with an !).
        - –stdin: This option is used to indicate that passwd should read the new password from standard input, which can be a pipe.
        - -u: This is the reverse of the -l option - it will unlock the account password by removing the !
        - -d: This is a quick way to delete a password for an account. It will set the named account passwordless. Available to root only.

- -e: This is a quick way to expire a password for an account. The user will be forced to change the password during the next login attempt. Available to root only.
- -n: This will set the minimum password lifetime, in days, if the user's account supports password lifetimes. Available to root only.
- -x: This will set the maximum password lifetime, in days, if the user's account supports password lifetimes. Available to root only.
- -w: This will set the number of days in advance the user will begin receiving warnings that her password will expire, if the user's account supports password lifetimes. Available to root only.
- -I: This will set the number of days which will pass before an expired password for this account will be taken to mean that the account is inactive and should be disabled, if the user's account supports password lifetimes. Available to root only.
- -S: This will output a short information about the status of the password for a given account. Available to root user only.

```
Changing password for kamal.
Current password:
New password:
Retype new password:
Bad: new password is just a wrapped version of the old one
New password:
Retype new password:
passwd: password updated successfully
kamal@kamal:~$
```

- **uname**
  - uname command is used to display basic information about the operating system and hardware. With options, Uname prints kernel details, and system architecture. Uname is the short name for 'UNIX name'. Uname command works on all Linux and Unix-like operating systems.
  - SYNTAX:

    uname [OPTION]
  - Options:
    - -a, --all: print all information, in the following order, except omit -p and -i if unknown:
    - -s, --kernel-name: print the kernel name
  - -n, --nodename: print the network node hostname
  - -r, --kernel-release: print the kernel release
  - -v, --kernel-version: print the kernel version
  - -m, --machine: print the machine hardware name

- -p, --processor: print the processor type or "unknown"
- -i, --hardware-platform: print the hardware platform or"unknown"
- -o, --operating-system: print the operating system
- –help: display this help and exit
- --version: output version information and exit

```
> uname -a
Darwin mac.local 21.6.0 Darwin Kernel Version 21.6.0:
Sat Jun 18 17:05:47 PDT 2022; root:xnu-8020.140.41~1/R
ELEASE_ARM64_T8101 arm64
> uname -s
Darwin
> uname -n
mac.local
> uname -v
Darwin Kernel Version 21.6.0: Sat Jun 18 17:05:47 PDT
2022; root:xnu-8020.140.41~1/RELEASE_ARM64_T8101
> uname -r
21.6.0
> uname -m
arm64
> uname -p
arm
> uname -i
uname: illegal option -- i
usage: uname [-amnprsv]
> uname -o
uname: illegal option -- o
usage: uname [-amnprsv]
```

uptime

- Uptime Command In Linux: It is used to find out how long the system is active (running). This command returns set of values that involve, the current time, and the amount of time system is in running state, number of users currently logged into, and the load time for the past 1, 5 and 15 minutes respectively.

- Syntax:

  uptime [-options]

- Options:

  - -p, --pretty show uptime in pretty format
  - -h, --help display this help and exit
  - -s, --since system up since
  - -V, --version output version information and exit

```
> uptime
13:32  up 2 days, 18:53, 1 user, load averages: 1.21 1.32 1.35
~ ------------------------------------------------------------
>
```

logname

- The Linux logname command is a simple utility that is part of the GNU Core Utilities. It has a single purpose, to print the name of the current user. There are no options and the command takes no arguments. You simply call it and it prints the current users login name.
- Syntax:

  $ logname

```
> logname
root
~ ----------------------------------------------------------
>
```

Calender

- If a user wants a quick view of the calendar in the Linux terminal, cal is the command for you. By default, the cal command shows the current month calendar as output.
- cal command is a calendar command in Linux which is used to see the calendar of a specific month or a whole year.
- Syntax:

  cal [[month] year]

```
> cal
   December 2022
Su Mo Tu We Th Fr Sa
                 1  2  3
 4  5  6  7  8  9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30 31

~ ------------------------------------------------
>
```

- **date**

  - date command is used to display the system date and time. date command is also used to set date and time of the system. By default the date command displays the date in the time zone on which unix/linux operating system is configured.You must be the super-user (root) to change the date and time.

  - Syntax:

    date [OPTION]... [+FORMAT]

    date [-u|--utc|--universal] [MMDDhhmm[[CC]YY][.ss]]

```
> date
Sat Dec 10 13:35:07 IST 2022
> date -u
Sat Dec 10 08:05:48 UTC 2022
~ ------------------------------------------------------------
>
```

- **hostname**
    - hostname command in Linux is used to obtain the DNS(Domain Name System) name and set the system's hostname or NIS(Network Information System) domain name. A hostname is a name which is given to a computer and it attached to the network. Its main purpose is to uniquely identify over a network.
    - Syntax :

        hostname -[option] [file]

```
> hostname
mac.local
~ ----------------------------------------
>
```

- **tty**
    - The tty command of terminal basically prints the file name of the terminal connected to standard input. tty is short of teletype, but popularly known as a terminal it allows you to interact with the system by passing on the data (you input) to the system, and displaying the output produced by the system.
    - Syntax:

      tty [OPTION]....

    - Options:
        - -s, --silent, --quiet: Prints nothing, only returns an exit status.
        - --help: It will display the help message and exit.
        - --version: Prints the version information and exits.

```
> tty
/dev/ttys000
~ ----------------------------------------
>
```

Lab 2

Directory command

- **pwd**

  - pwd stands for Print Working Directory. It prints the path of the working directory, starting from the root.
  - pwd is shell built-in command(pwd) or an actual binary(/bin/pwd).
  - Syntax:

    pwd [-options]

  - Options:

    - pwd -L: Prints the symbolic path.
    - pwd -P: Prints the actual path.

```
> pwd
/Users/apoorv/desktop/SEM5
~/desktop/SEM5 main !98 ?3 ----------------------
>
```

- **cd**

  - cd command in linux known as change directory command. It is used to change current working directory.

  - Syntax:

    $ cd [directory]

```
> cd theory
~/desktop/SEM5/theory main !98 ?3 ----------
>
```

- **mkdir**

  - mkdir command in Linux allows the user to create directories (also referred to as folders in some operating systems). This command can create multiple directories at once as well as set the permissions for the directories. It is important to note that the user executing this command must have enough permissions to create a directory in the parent directory, or he/she may receive a 'permission denied' error.

  - Syntax:

    mkdir [options...] [directories ...]

  - Options:

    - –version: It displays the version number, some information regarding the license and exits.
    - –help: It displays the help related information and exits.
    - -v or –verbose: It displays a message for every directory created.
    - -p: A flag which enables the command to create parent directories as necessary. If the directories exist, no error is specified.
    - -m: This option is used to set the file modes, i.e, permissions, etc. for the created directories

```
> mkdir directory1
> mkdir directory2
~/Desktop main +6 !3 ?2 ------------------------
>
```

- **rmdir**

  - rmdir command is used remove empty directories from the filesystem in Linux. The rmdir command removes each and every directory specified in the command line only if these directories are empty. So if the specified directory has some directories or files in it then this cannot be removed by rmdir command.

  - Syntax:

    rmdir [-p] [-v | –verbose] [–ignore-fail-on-non-empty] directories ...

  - Options:

    - –help: It will print the general syntax of the command along with the various options that can be used with the rmdir command as well as give a brief description about each option.
    - rmdir -p: In this option each of the directory argument is treated as a pathname of which all components will be removed, if they are already empty, starting from the last component.
    - rmdir -v, –verbose: This option displays verbose information for every directory being processed.
    - rmdir –ignore-fail-on-non-empty: This option do not report a failure which occurs solely because a directory is non-empty. Normally, when rmdir is being instructed to remove a non-empty directory, it simply reports an error. This option consists of all those error messages.

- rmdir –version: This option is used to display the version information and exit.

```
> rmdir dir1
rmdir: dir1: No such file or directory
> rmdir directory1
~/Desktop main +6 !3 ?2 ----------------------------------
>
```

Lab 3

File commands

- **cat**
  - Cat(concatenate) command is very frequently used in Linux. It reads data from the file and gives their content as output. It helps us to create, view, concatenate files. So let us see some frequently used cat commands.
  - Syntax:

    $ cat [OPTION] [FILE]...

```
> cat f00
This is Apoorv Aron
```

- **cp**

  - cp stands for copy. This command is used to copy files or group of files or directory. It creates an exact image of a file on a disk with different file name. cp command requires at least two filenames in its arguments.

  - Syntax:

    cp [OPTION] Source Destination

    cp [OPTION] Source Directory

    cp [OPTION] Source-1 Source-2 Source-3 Source-n Directory

```
> cp f00 f11
> cat f11
This is Apoorv Aron
~/Desktop main +6 !3 ?2 ------------
>
```

- **Copy**

  - cp stands for copy. This command is used to copy files or group of files or directory. It creates an exact image of a file on a disk with different file name. cp command requires at least two filenames in its arguments.

  - Syntax:

    cp [OPTION] Source Destination

    cp [OPTION] Source Directory

    cp [OPTION] Source-1 Source-2 Source-3 Source-n Directory

```
> cp f00 f11
> cat f11
This is Apoorv Aron
~/Desktop main +6 !3 ?2 ----------
>
```

- **rm**

  - rm stands for remove here. rm command is used to remove objects such as files, directories, symbolic links and so on from the file system like UNIX. To be more precise, rm removes references to objects from the filesystem, where those objects might have had multiple references (for example, a file with two different names). By default, it does not remove directories.

  - Syntax:

    rm [OPTION]... FILE...

```
> rm f11
> ls
SEM5 f00  git
~/Desktop main +6 !3 ?1 ----------------------
>
```

- **wc**

  - wc stands for word count. As the name implies, it is mainly used for counting purpose.
  - It is used to find out number of lines, word count, byte and characters count in the files specified in the file arguments.
  - By default it displays four-columnar output.
  - First column shows number of lines present in a file specified, second column shows number of words present in the file, third column shows number of characters present in file and fourth column itself is the file name which are given as argument.
  - Syntax:

    wc [OPTION]... [FILE]...

```
> wc f00
       2       6      30 f00
> wc -l f00
       2 f00
> wc -w f00
       6 f00
~/Desktop main +6 !3 ?1 ----------------------------
>
```

- **cmp**

  - cmp command in Linux/UNIX is used to compare the two files byte by byte and helps you to find out whether the two files are identical or not.

  - When cmp is used for comparison between two files, it reports the location of the first mismatch to the screen if difference is found and if no difference is found i.e the files compared are identical.

  - cmp displays no message and simply returns the prompt if the the files compared are identical.

  - Syntax:

    cmp [OPTION]... FILE1 [FILE2 [SKIP1 [SKIP2]]]

```
> cat f00
This is Apoorv Aron
> cat f11
This is Apoorv
> cmp f00 f11
f00 f11 differ: char 15, line 1
~/Desktop main +6 !3 ?2 ---------------
>
```

- **diff**

  - o diff stands for difference. This command is used to display the differences in the files by comparing the files line by line. Unlike its fellow members, cmp and comm, it tells us which lines in one file have is to be changed to make the two files identical.

  - o Special symbols are:

  - o a : add

  - o c : change

  - o d : delete

  - o Syntax :

    diff [options] File1 File2

```
> diff f00 f11
1c1
< This is Apoorv Aron
---
> This is Apoorv
~/Desktop main +6 !3 ?2 ----------
>
```

- **comm**

  - comm compare two sorted files line by line and write to standard output; the lines that are common and the lines that are unique.

  - Syntax:

    $comm [OPTION]... FILE1 FILE2

  - Options for comm command:

    - -1 :suppress first column(lines unique to first file).
    - -2 :suppress second column(lines unique to second file).
    - -3 :suppress third column(lines common to both files).
    - – -check-order :check that the input is correctly sorted, even if all input lines are pairable.
    - – -nocheck-order :do not check that the input is correctly sorted.
    - – -output-delimiter=STR :separate columns with string STR
    - – -help :display a help message, and exit.
    - – -version :output version information, and exit.

```
> comm f00 f11
        This is Apoorv
This is Apoorv Aron
~/Desktop main +6 !3 ?2 ----------------------
>
```

Lab 4

List command

- **ls**

  - ls is a Linux shell command that lists directory contents of files and directories

  - Syntax

    ls

```
> ls
Drive Link.pages     Student_Details.xlsx acad_calendar.jpeg   paper
IT_syllabus.pdf      Theory               btech_mtech.pdf      practical
~/Desktop/sem5 main !100 ?5 ------------------------------------------------
>
```

- **ls -a**

  - To show all the hidden files in the directory, use '-a option'.
  - Syntax:

    ls -a

```
> ls -a
.                   .git                Student_Details.xlsx btech_mtech.pdf
..                  Drive Link.pages    Theory              paper
.DS_Store           IT_syllabus.pdf     acad_calendar.jpeg  practical
~/Desktop/sem5 main !100 ?5 ----------------------------------------------
>
```

- **ls -l**

  - Will get the details of directories content.
  - Syntax:

    ls -l

```
> ls -l
total 2568
-rwx------@  1 apoorv  staff  104605 Feb  6  2022 Drive Link.pages
-rw-r--r--@  1 apoorv  staff  435919 Mar  9  2022 IT_syllabus.pdf
-rw-r--r--@  1 apoorv  staff   11800 Mar 14  2022 Student_Details.xlsx
drwxr-xr-x   3 apoorv  staff      96 Jul  5 15:20 Theory
-rw-r--r--@  1 apoorv  staff  282449 Dec  2 21:27 acad_calendar.jpeg
-rw-r--r--@  1 apoorv  staff  471042 Aug 12 19:26 btech_mtech.pdf
drwxr-xr-x  26 apoorv  staff     832 Dec  2 21:23 paper
drwxr-xr-x   3 apoorv  staff      96 Jul  5 15:20 practical
~/Desktop/sem5 main !100 ?5 ----------------------------------------------
>
```

Lab 5
Permissions

- **chmod**

The chmod command is used to change the access mode of a file.

Syntax : chmod [reference][operator][mode] file...

Reference	Class	Description
u	owner	File's owner
g	group	Users who are the member of the file's group
o	others	Users who are neither the file's owner nor members of the file's group
a	all	All three of the above

Operator	Description
+	Adds the specified modes to the specified classes
-	Removes the specified modes from the specified classes
=	The modes specified are to be made the exact modes for the specified classes

r	Permission to read the file
w	Permission to write (or delete) the file
x	Permission to execute the file, or, in the case of a director search it

```
> cd ~/Desktop/
> touch f00
> ls -l f00
-rw-r--r--  1 apoorv  staff  0 Dec 10 14:06 f00
> chmod u+x f00
> ls -l f00
-rwxr--r--  1 apoorv  staff  0 Dec 10 14:06 f00
~/Desktop main +6 !3 ?1 ----------------------------------------
>
```

- **chgrp**

chgrp command in Linux is used to change the group ownership of afile or directory. All files in Linux belong to an owner and a group.

Syntax : chgrp [OPTION]… GROUP FILE…
chgrp [OPTION]… –reference=RFILE FILE…

```
kamal@kamal:~$ ls -l file
-rwxrwxrwx 1 kamal kamal 0 Dec 16 23:13 file
kamal@kamal:~$ sudo addgroup linux_file
[sudo] password for kamal:
Adding group `linux_file' (GID 1003) ...
Done.
kamal@kamal:~$ sudo chgrp linux_lab file
chgrp: invalid group: ‘linux_lab’
kamal@kamal:~$ sudo chgrp linux_file file
kamal@kamal:~$ ls -l file
-rwxrwxrwx 1 kamal linux_file 0 Dec 16 23:13 file
kamal@kamal:~$
```

- **chown**

The chown command allows you to change the user and/or group ownership of a given file, directory, or symbolic link.

In Linux, all files are associated with an owner and a group and assigned with permission access rights for the file owner, the group members, and others

Syntax : chown [OPTION]... [OWNER][:[GROUP]] FILE...
chown [OPTION]... –reference=RFILE FILE...

Types of file Permissions:

- User: These types of file permissions affect the owner of the file.
- Group: These types of file permissions affect the group which owns the file. Instead of the group permissions, the user permissions will apply if the owner user is in this group.
- Other: These types of file permissions affect all other users on the system.

```
> ls -l f00
-rwxr--r--  1 apoorv  staff  0 Dec 10 14:06 f00
> sudo chown apoorv
Password:
Sorry, try again.
Password:
usage: chown [-fhnvx] [-R [-H | -L | -P]] owner[:group] file ...
       chown [-fhnvx] [-R [-H | -L | -P]] :group file ...
```

Lab 6
Process Commands

- **ps**

  Linux provides us a utility called ps for viewing information related with the processes on a system which stands as abbreviation for "Process Status". ps command is used to list the currently running processes and their PIDs along with some other information depends on different options. It reads the process information from the virtual files in /proc file-system. /proc contains virtual files, this is the reason it's referred to as a virtual file system.

  Syntax: ps [options]

  Options

-A, -e	All except the session leaders
-a	All with tty, except session leaders.
A	All with tty including other users.
-d	All except sessions leaders
-N, --deselect	Negate selection
r	Only running processes
T	All processes on this terminal

x	Processes without controlling ttysps

```
> ps
  PID TTY           TIME CMD
86093 ttys000    0:00.41 /opt/homebrew/bin/zsh -il
86131 ttys000    0:00.00 /opt/homebrew/bin/zsh -il
86182 ttys000    0:00.00 /opt/homebrew/bin/zsh -il
86184 ttys000    0:00.03 /opt/homebrew/bin/zsh -il
86185 ttys000    0:00.14 /Users/apoorv/.cache/gitstatus/gitstatusd-darwin-arm64 -G v1.3.1 -s -1 -u -1 -d -1 -
42735 ttys001    0:03.46 /bin/zsh
42781 ttys001    0:00.00 /bin/zsh
42782 ttys001    0:00.27 /bin/zsh
42785 ttys001    0:00.00 /bin/zsh
42790 ttys001    0:01.07 /Users/apoorv/.cache/gitstatus/gitstatusd-darwin-arm64 -G v1.3.1 -s -1 -u -1 -d -1 -
```

```
> ps x
  PID   TT  STAT      TIME COMMAND
  532   ??  S      1:14.25 /usr/sbin/distnoted agent
  556   ??  S      0:43.38 /usr/libexec/UserEventAgent (Aqua)
  562   ??  S      0:56.79 /System/Library/CoreServices/Dock.app/Contents/MacOS/Dock
  563   ??  S      2:46.06 /System/Library/CoreServices/ControlCenter.app/Contents/MacOS/ControlCenter
  564   ??  S      0:06.68 /System/Library/CoreServices/SystemUIServer.app/Contents/MacOS/SystemUIServer
  565   ??  S      0:22.42 /usr/libexec/lsd
  566   ??  S      4:30.47 /System/Library/CoreServices/Finder.app/Contents/MacOS/Finder
  573   ??  S      0:18.33 /usr/libexec/nsurlsessiond
  574   ??  S      0:22.45 /System/Library/CoreServices/lockoutagent
  575   ??  S      0:33.91 /usr/sbin/usernoted
  582   ??  S      0:16.76 /usr/libexec/rapportd
  585   ??  S      0:08.21 /System/Library/PrivateFrameworks/TelephonyUtilities.framework/callservicesd
  589   ??  S      0:09.01 /System/Library/PrivateFrameworks/IDS.framework/identityservicesd.app/Contents/Mac
  590   ??  S      0:52.84 /System/Library/Frameworks/Accounts.framework/Versions/A/Support/accountsd
  592   ??  S      0:01.93 /usr/libexec/pboard
  596   ??  S      7:56.92 /System/Library/PrivateFrameworks/CalendarAgent.framework/Executables/CalendarAgen
  598   ??  S      0:02.71 /System/Library/PrivateFrameworks/CloudDocsDaemon.framework/Versions/A/Support/bir
  599   ??  S      0:06.67 /System/Library/PrivateFrameworks/iTunesCloud.framework/Support/itunescloudd
  600   ??  S      0:10.35 /System/Library/PrivateFrameworks/ViewBridge.framework/Versions/A/XPCServices/View
```

```
> ps a
  PID   TT  STAT      TIME COMMAND
86093 s000  Ss+    0:00.42 /opt/homebrew/bin/zsh -il
86131 s000  S      0:00.00 /opt/homebrew/bin/zsh -il
86182 s000  S      0:00.00 /opt/homebrew/bin/zsh -il
86184 s000  S      0:00.03 /opt/homebrew/bin/zsh -il
86185 s000  S      0:00.14 /Users/apoorv/.cache/gitstatus/gitstatusd-darwin-arm64 -G v1.3.1 -s -1 -u -1 -d -1
42735 s001  Ss     0:03.61 /bin/zsh
42781 s001  S      0:00.00 /bin/zsh
42782 s001  S      0:00.27 /bin/zsh
42785 s001  S      0:00.00 /bin/zsh
42790 s001  S      0:01.07 /Users/apoorv/.cache/gitstatus/gitstatusd-darwin-arm64 -G v1.3.1 -s -1 -u -1 -d -1
92411 s001  R+     0:00.00 ps a
```

- **top**

  top command is used to show the Linux processes. It provides a dynamic real-time view of the running system. Usually, this command shows the summary information of the system and the list of processes or threads which are currently managed by the Linux Kernel.

  Syntax : top [options]

  Pressing q will exit

```
Processes: 344 total, 4 running, 340 sleeping, 2239 threads                                    20:09:29
Load Avg: 2.02, 2.02, 2.01  CPU usage: 17.87% user, 5.7% sys, 77.5% idle
SharedLibs: 286M resident, 59M data, 17M linkedit.
MemRegions: 476252 total, 1765M resident, 98M private, 1225M shared.
PhysMem: 7589M used (1365M wired), 61M unused.
VM: 168T vsize, 3831M framework vsize, 44757808(4) swapins, 50875492(0) swapouts.
Networks: packets: 12393929/13G in, 5005569/2467M out. Disks: 24937926/1117G read, 21865128/970G written.

PID    COMMAND      %CPU  TIME     #TH   #WQ  #PORT MEM    PURG   CMPRS  PGRP  PPID  STATE    BOOSTS
85614  Google Chrom 107.9 18:11.56 35/1  1    359   298M+  0B     74M+   1054  1054  running  *0[6]
379    WindowServer 22.0  04:17:27 18    6    2049- 677M+  21M-   271M-  379   1     sleeping *0[1]
0      kernel_task  13.2  03:17:46 460/8 0    0     61M    0B     0B     0     0     running   0[0]
77162  Discord Help 6.2   80:09.36 35    2    364   169M+  0B     123M   77118 77118 sleeping *0[12]
77122  Discord Help 5.8   71:51.08 11    3    174   119M   0B     34M    77118 77118 sleeping *1[11]
369    bluetoothd   4.9   24:47.50 11    3    281   9937K  0B     3936K  369   1     sleeping *0[1]
424    coreaudiod   3.8   62:35.08 16    6    909-  31M-   0B     16M    424   1     sleeping *0[1]
9885   iTerm2       3.4   02:50.30 8     5    286   128M+  64M    37M-   9885  1     sleeping *0[6356]
92658  top          3.1   00:01.03 1/1   0    27    5377K+ 0B     0B     92658 42735 running  *0[1]
92671  screencaptur 2.3   00:00.29 6     4    143   8594K  752K   0B     564   564   sleeping *0[214+]
1120   Google Chrom 2.2   44:58.82 14    2    197   18M    0B     12M    1054  1054  sleeping *0[4]
```

Highlight running process in top:

```
> top -h
top usage: top
                 [-a | -d | -e | -c <mode>]
                 [-F | -f]
                 [-h]
                 [-i <interval>]
                 [-l <samples>]
                 [-ncols <columns>]
                 [-o <key>] [-O <secondaryKey>]
                         keys: pid (default), command, cpu, cpu_me, cpu_others, csw,
                                 time, threads, ports, mregion, mem, rprvt, purg, vsize, vprvt,
                                 kprvt, kshrd, pgrp, ppid, state, uid, wq, faults, cow, user,
                                 msgsent, msgrecv, sysbsd, sysmach, pageins, boosts, instrs, cycles
                 [-R | -r]
                 [-S]
                 [-s <delay>]
                 [-n <nprocs>]
                 [-stats <key(s)>]
                 [-pid <processid>]
                 [-user <username>]
                 [-U <username>]
                 [-u]
```

- **bg**

  bg command in linux is used to place mentioned foreground jobs in background.

  Syntax : bg [job_spec…]

```
> jobs
> sleep 500
^Z
[1]  + 92909 suspended  sleep 500
> bg
[1]  + 92909 continued  sleep 500
```

- **fg**

fg command in linux is used to place background jobs in foreground.

Syntax: fg [jobs_spec…]

```
> jobs
> sleep 500
^Z
[1]  + 92909 suspended  sleep 500
> bg
[1]  + 92909 continued  sleep 500
> FG
/usr/bin/FG: line 4: fg: no job control
```

- **nice**

  nice command in Linux helps in execution of a program/process with modified scheduling priority**.**

  Syntax : nice [Option][Command][ARG]…]

```
> ps -el | grep terminal
  501 93317 42735     4006   0  31  0 408636560   1680 -      S+                0 ttys001    0:00.00 grep --
color=auto --exclude-dir=.bzr --exclude-dir=CVS --exclude-dir=.git --exclude-dir=.hg --exclude-dir=.svn --excl
ude-dir=.idea --exclude-dir=.tox terminal
```

As soon as we change priority of terminal, a new terminal opens

- **renice:**
  Renice command is used to change priority of running process using id

  Syntax: renice -n [priotiy] [

```
> nice -10 gnome-terminal
nice: gnome-terminal: No such file or directory
> renice -n 15 -p 5006
renice: 5006: getpriority: No such process
```

- **Kill:** ***kill*** command in Linux (located in /bin/kill), is a built-in command which is used to terminate processes manually. ***kill*** command sends a signal to a process which terminates the process.

  Syntax : kill [PID]

```
> ps
  PID TTY           TIME CMD
86093 ttys000    0:00.48 /opt/homebrew/bin/zsh -il
86131 ttys000    0:00.00 /opt/homebrew/bin/zsh -il
86182 ttys000    0:00.00 /opt/homebrew/bin/zsh -il
86184 ttys000    0:00.04 /opt/homebrew/bin/zsh -il
86185 ttys000    0:00.16 /Users/apoorv/.cache/gitstatus/gitstatusd-darwin-arm64 -G v1.3.1 -s -1 -u -1 -d -1 -
42735 ttys001    0:06.07 /bin/zsh
42781 ttys001    0:00.00 /bin/zsh
42782 ttys001    0:00.27 /bin/zsh
42785 ttys001    0:00.00 /bin/zsh
42790 ttys001    0:01.12 /Users/apoorv/.cache/gitstatus/gitstatusd-darwin-arm64 -G v1.3.1 -s -1 -u -1 -d -1 -
> kill 86093
```

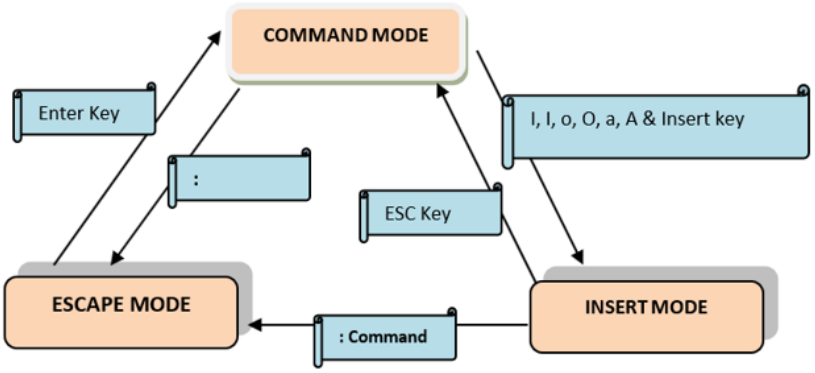
Lab 10

VI Editor

VI Editor: The vi editor is elaborated as **vi**sual editor. It is installed in every Unix system. In other words, it is available in all Linux distros. It is user-friendly and works same on different distros and platforms. It is a very powerful application. An improved version of vi editor is **vim**.

The vi-editor tool is an interactive tool as it displays changes made in the file on the screen while you edit the file. In vi editor you can insert, edit or remove a word as cursor moves throughout the file.

Modes of VI Editor:



While working with the vi editor, we usually come across the following modes:

- **Command mode:** This mode enables you to perform administrative tasks such as saving the files, executing the commands etc. In this mode, whatever you type is interpreted as a command.
- **Insert mode:** This mode enables you to insert text into the file. Everything that's typed in this mode is interpreted as input and placed in the file.
- VI always starts in the command mode. To enter text, you must be in the insert mode for which simply type
- To come out of the insert mode, press the Esc key, which will take you back to the command mode.

Following are some basic commands to use vi editor:

- vi [filename]: Creates a new file if does not exists already, otherwise opens the existing file.

- vi -R: Opens a file in read-only mode.

  A tilde (~) on each line represents an unused line. If a line does not begin with a tilde and appears to be blank, there is a space, tab, newline, or some other non-viewable character present.

VI editing commands:

- i : Insert at current cursor position.
- a : Write after cursor.
- A : Write at end of line.
- ESC : Terminate insert mode.
- u : Undo last change.
- U : Undo all changes to the entire line.
- o : Open a new line (goes into insert mode).
- dd : Delete line.
- D : Delete contents of line after the cursor
- C : Delete contents of a line after the cursor and insert new text.
- dw : Delete word.
- cw : Change word.
- x : Delete character at the cursor.
- r : Replace character

Moving within a file:

- k : Move cursor up.
- j : Move cursor down.
- h : Move cursor left.
- l : Move cursor right.

Saving & Closing file:

- Shift+zz : Save the file and quit.
- :w : Save the file but keep it open.
- :q : Quit without saving.
- :wq : Save the file and quit

Lab 11
WSS to enter two strings

```
#!/bin/bash

# Prompt the user to enter the first string
echo "Enter the first string: "
read string1

# Prompt the user to enter the second string
echo "Enter the second string: "
read string2

echo "The first string is: $string1"
echo "The second string is: $string2"
```

```
> vi prac10.sh
> bash ./prac10.sh
Enter the first string:
This is 11th practical
Enter the second string:
Next is 12th
The first string is: This is 11th practical
The second string is: Next is 12th
```

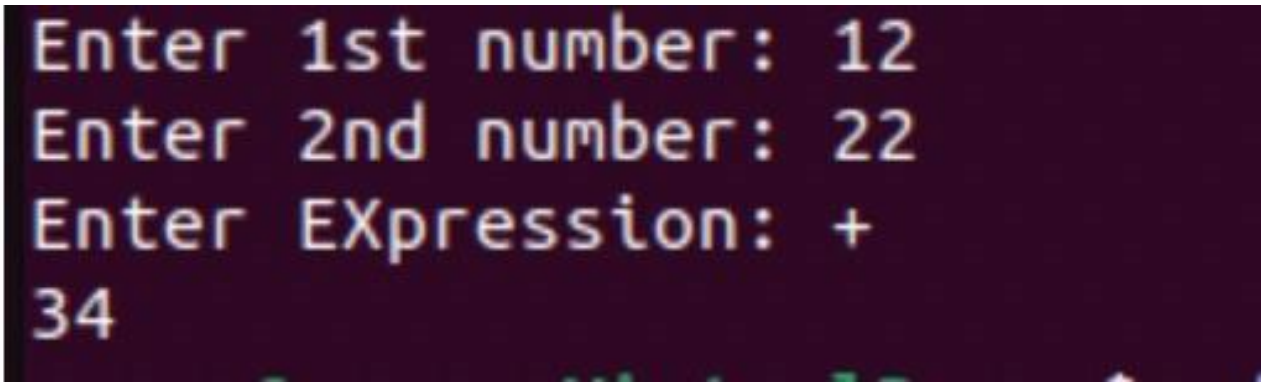
Lab 12

WSS to add, subtract, multiply and divide two numbers.

```
#!/bin/bash
#Taking the inputs from user through prompt
read -p "Enter the first number: " num1
read -p "Enter the second number: " num2

#using expr and read to take operator
read -p “Enter expression: “ expr

#Using bc for calculation
echo “The result is: “ $num1$expr$num2 | bc
```



Lab 13
WSS to find largest of two number

```
#! /bin/bash

read -p 'Enter the first number: ' num1
read -p 'Enter the second number: ' num2

if [ $num1 -gt $num2]
then
        echo 'The larger number is: ' $num1
else
        echo 'Then larger number is: ' $num2
fi
```

```
> vi prac13.sh
> bash ./prac13.sh
Enter Num1
44
Enter Num2
55
55
```

Lab 14
WSS to Check whether given number is even or odd.

```
#!/bin/bash

# Prompt the user to enter a number
echo "Enter a number: "
read number

# Check if the number is even or odd
if [ $(($number % 2)) -eq 0 ]
then
  echo "$number is even"
else
  echo "$number is odd"
fi
```

```
> vi prac14.sh
> bash ./prac14.sh
Enter a number:
41
41 is odd
> bash ./prac14.sh
Enter a number:
42
42 is even
```

Lab 15
WSS to list all the directory files in a directory

Source code:

```
echo "ls command can be run by shell script"

ls -l
~
~
```

Output:

```
> vi prac15.sh
> bash ./prac15.sh
ls command can be run by shell script
total 16
drwxr-xr-x  12 apoorv  staff  384 Dec 10 14:05 SEM5
drwxr-xr-x  11 apoorv  staff  352 Dec  2 21:19 git
drwxr-xr-x   7 apoorv  staff  224 Dec 30 14:07 node
-rw-r--r--   1 apoorv  staff  211 Dec 30 20:34 prac14.sh
-rw-r--r--   1 apoorv  staff   52 Dec 30 20:38 prac15.sh
~/Desktop main +6 !3 ?3 ----------------------------------------
>
```

Lab 16
WSS to find factorial of a number

Source Code:

```
# shell script for factorial of a number
# factorial using for loop

echo "Enter a number"

# Read the number
read num

fact=1

for((i=2;i<=num;i++))
{
  fact=$((fact * i))
}

echo $fact
~
```

Output:

```
> vi prac16.sh
> bash ./prac16.sh
Enter a number
4
24
~/Desktop main +6 !3 ?4 ----------
>
```

Lab 17
Write a c program to design a calculator

Source Code:

```
# !/bin/bash

# Take user Input
echo "Enter Two numbers : "
read a
read b

# Input type of operation
echo "Enter Choice :"
echo "1. Addition"
echo "2. Subtraction"
echo "3. Multiplication"
echo "4. Division"
read ch

# Switch Case to perform
# calculator operations
case $ch in
1)res=`echo $a + $b | bc`
;;
2)res=`echo $a - $b | bc`
;;
3)res=`echo $a \* $b | bc`
;;
4)res=`echo "scale=2; $a / $b" | bc`
;;
esac
echo "Result : $res"
```

Output:

```
> vi prac17.sh
> bash ./prac17.sh
Enter Two numbers :
34
2
Enter Choice :
1. Addition
2. Subtraction
3. Multiplication
4. Division
1
Result : 36
```

```
> vi prac17.sh
> bash ./prac17.sh
Enter Two numbers :
34
2
Enter Choice :
1. Addition
2. Subtraction
3. Multiplication
4. Division
4
Result : 17.00
~/Desktop main +6 !3 ?2 ------------------------------
>
```

Lab 18

Write a c program to create a child process and allow the parent to display “parent” and the child to display “child” on the screen.?

Source Code:

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

int main()
{
        pid_t pid;
        //create a child process
        pid = fork();

        if (pid < 0){
                //fork failed
                printf( "Fork Failed.\n");
                return 1;
        }
        else if (pid == 0){
        //This is child process
                printf("Child\n");
        }else {
        //This is parent process
                printf("Parent\n");
        }
        return 0;
}
~
~
```

Output:

```
~$ gcc prac18.c -o prac18
~$ ./prac18
Parent
```