Lab – 02

Task 1:

Text Editors: A text editor is a software application that allows users to create, edit, and manipulate plain text files.

Five Text Editors:

1. Vim
2. Nano
3. Gedit
4. Emacs
5. Sublime Text

Different Modes:

1- Normal mode: This is the default mode of Vim, where you can navigate and manipulate text using various commands. In normal mode, you can use keys to move the cursor, delete text, and perform other actions.

2- Insert mode: In this mode, you can insert new text into the file. You can enter insert mode by pressing the i key in normal mode. In insert mode, Vim behaves like a traditional text editor, allowing you to type and edit text freely.

3- Command mode: In this mode, you can enter commands to perform specific actions, such as saving the file, quitting Vim, or executing shell commands. You can enter command mode by pressing the : key in normal mode.

Task 2:

cp /etc/passwd practice.txt; cp /etc/passwd practice2.txt

a- To move to the beginning of the file-> ^ or 0, and end of the file by $

b- Move to the beginning of the file-> :0 and end to the file :$

c- Move to line 20-> 20G

b-

- go to the first line by gg and then dd, or simply 1dd, or :1d.
- 7dd or :7d, go to line 15 by 15G, reach the end of line by entering $, then enter "a" and type "ADD DYNAMIC".

c-

- gg followed by /usr, then enter cw , and type USER, then finally esc.
- :%s/usr/USER/g

d-

- :wq

e-

- vi + practice.txt
- vi +21 practice.txt
- vi +/USER practice.txt

f-

- Open files in horizontal split: vi -o practice.txt practice2.txt
  and in vertical split by: vi -O practice.txt practice2.txt
- Move the cursor between files by ctrl+w followed by w.
- enter v or V to select text and y to copy. To paste it in the other file go to the desired location and enter p.
- :wqa

Task 3:

- vi hello.c
- Type the code
  ```
  #include<stdio.h>
  int main(int argc, char** argv)
  {
  printf("Hello World\n");
  }
  ```
- :!gcc -save-temps -o hello hello.c
- :sh
  ```
  ./hello
  exit
  ```

Task 4:

- Preprocessed code file-> gcc -E linux_fun.c -o linux_fun.i
- Assembly code file -> gcc -S linux_fun.i -o linux_fun.s
- Object file-> gcc -c linux_fun.s -o linux_fun.o
- Executable file-> gcc -o linux_fun linux_fun.o

Task 5:

- Dynamic: gcc -o dynamic file.c
- Static: gcc - static -o static file.c

Difference in sizes:
The statically linked “static” executable will be much larger than the dynamically linked “dynamic” executable. This is because the statically linked executable includes all the necessary libraries and code, whereas the dynamically linked executable relies on external libraries.

To see their sizes:
ls -s | grep -E "static|dynamic"

Strace:
The main difference between the two outputs on doing strace is the presence of additional mmap system calls in the dynamically linked “dynamic” executable. These mmap calls are used to map the dynamic libraries into memory, which is not necessary for the statically linked “static” executable. dynamically linked binaries rely on the dynamic linker (usually ld-linux.so) to load the required libraries at runtime. The dynamic linker uses the openat syscall to open the library files and map them into memory.

Task 6:

1- Disassembly: objdump -d -M intel cat_dynamic
2- ELF Header: readelf -h dynamic
3- Program Header: readelf -l dynamic
4- Section Header: readelf -S cat_dynamic

Task 7:

- Ctrl+z to stop a process
- Ctrl+c to kill a process
- job is used to display information about the current jobs.
- fg is used to bring a job to the foreground
- bg is used to run a job in the background
- stop is used to stop a job. When you run a command with stop, it will suspend the job and you can resume it later.