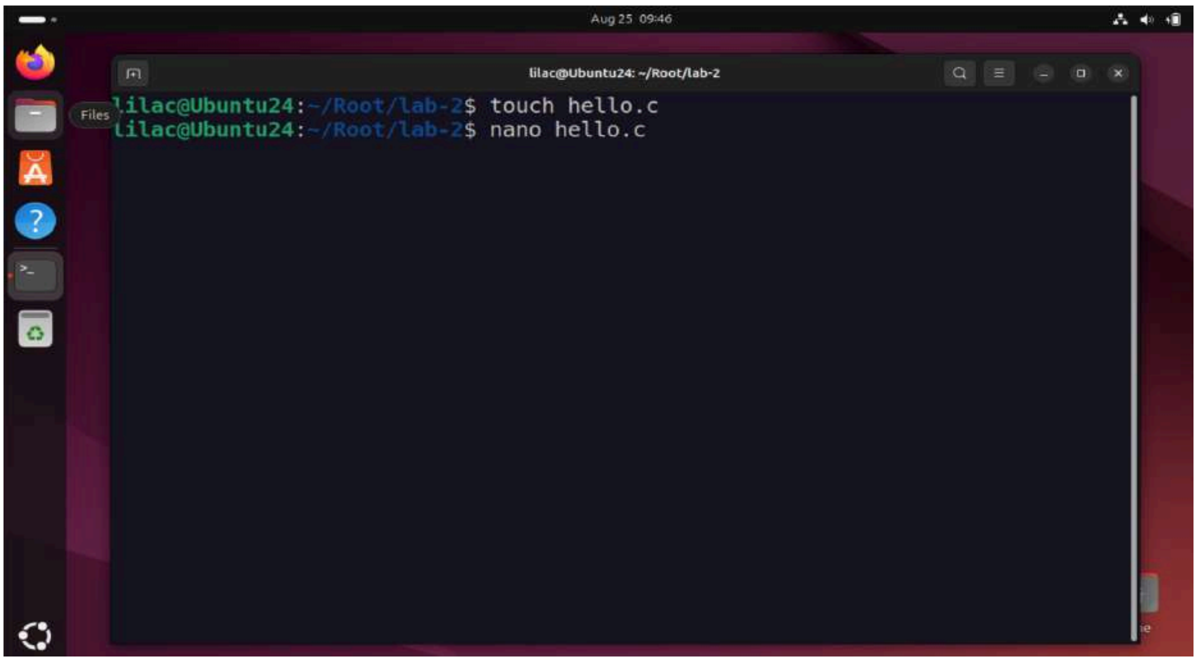
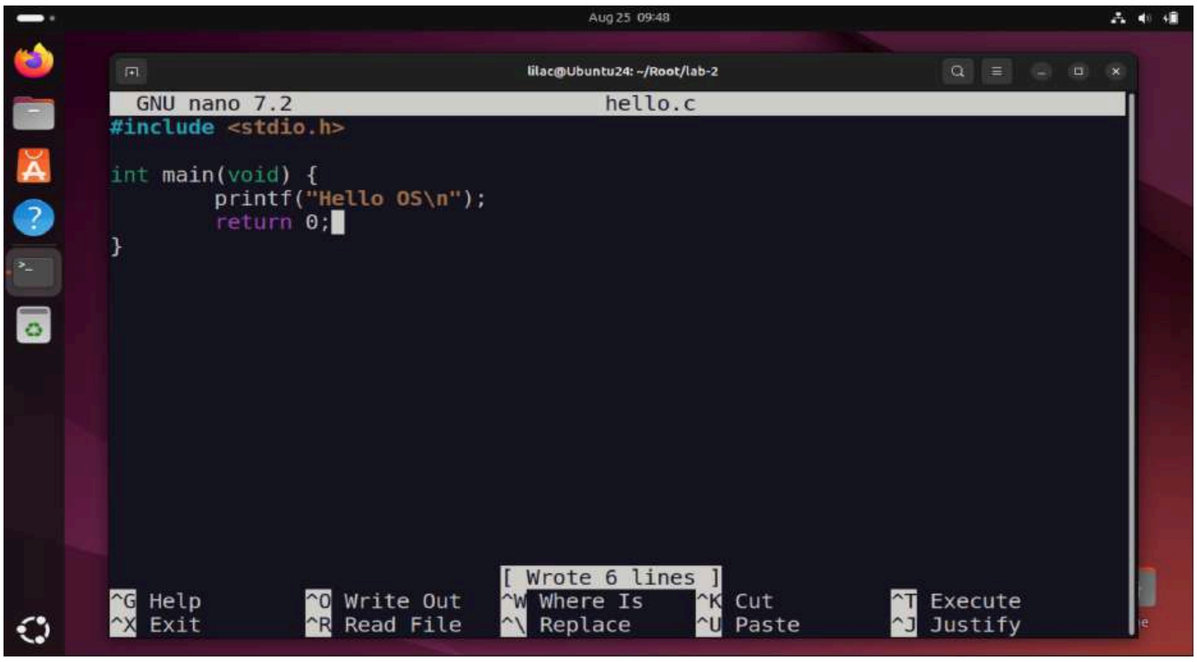
Michael Xavier E. Canonizado
BSCS 3A
Operating Systems - Lab 2

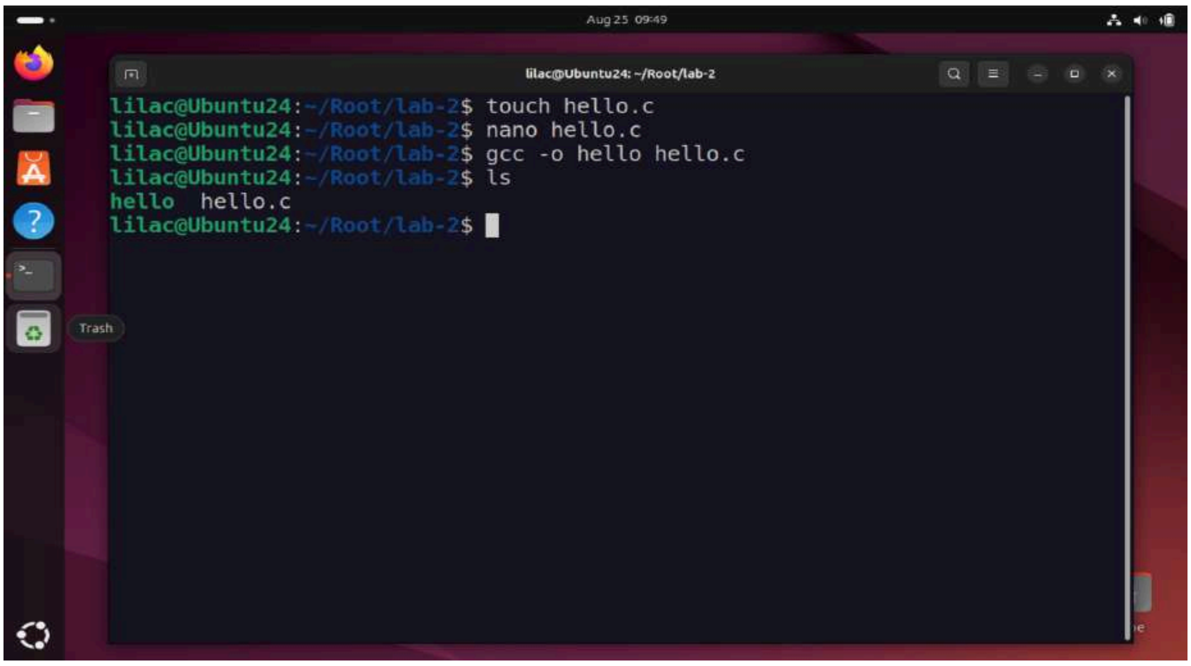
1.



```
lilac@Ubuntu24:~/Root/lab-2$ touch hello.c
lilac@Ubuntu24:~/Root/lab-2$ nano hello.c
```

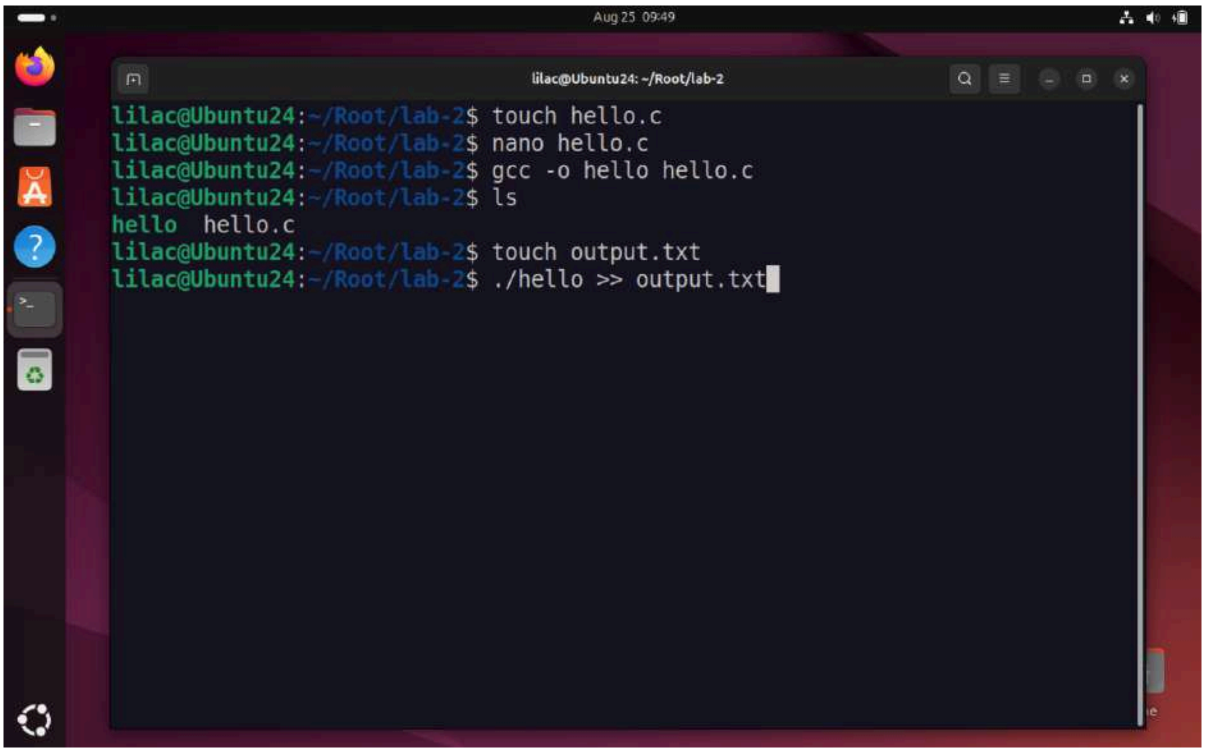


```
  GNU nano 7.2                        hello.c
#include <stdio.h>

int main(void) {
        printf("Hello OS\n");
        return 0;
}

                        [ Wrote 6 lines ]
^G Help        ^O Write Out   ^W Where Is    ^K Cut         ^T Execute
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify
```

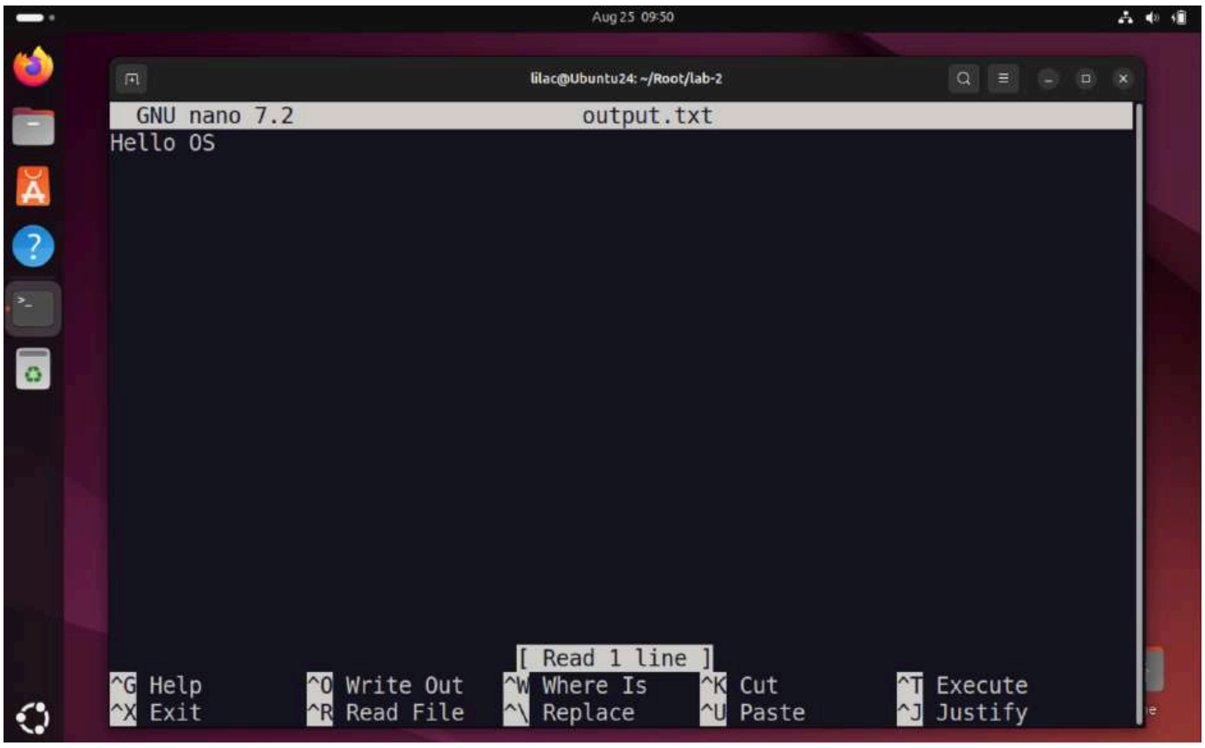


```
lilac@Ubuntu24:~/Root/lab-2$ touch hello.c
lilac@Ubuntu24:~/Root/lab-2$ nano hello.c
lilac@Ubuntu24:~/Root/lab-2$ gcc -o hello hello.c
lilac@Ubuntu24:~/Root/lab-2$ ls
hello  hello.c
lilac@Ubuntu24:~/Root/lab-2$ 
```

2.

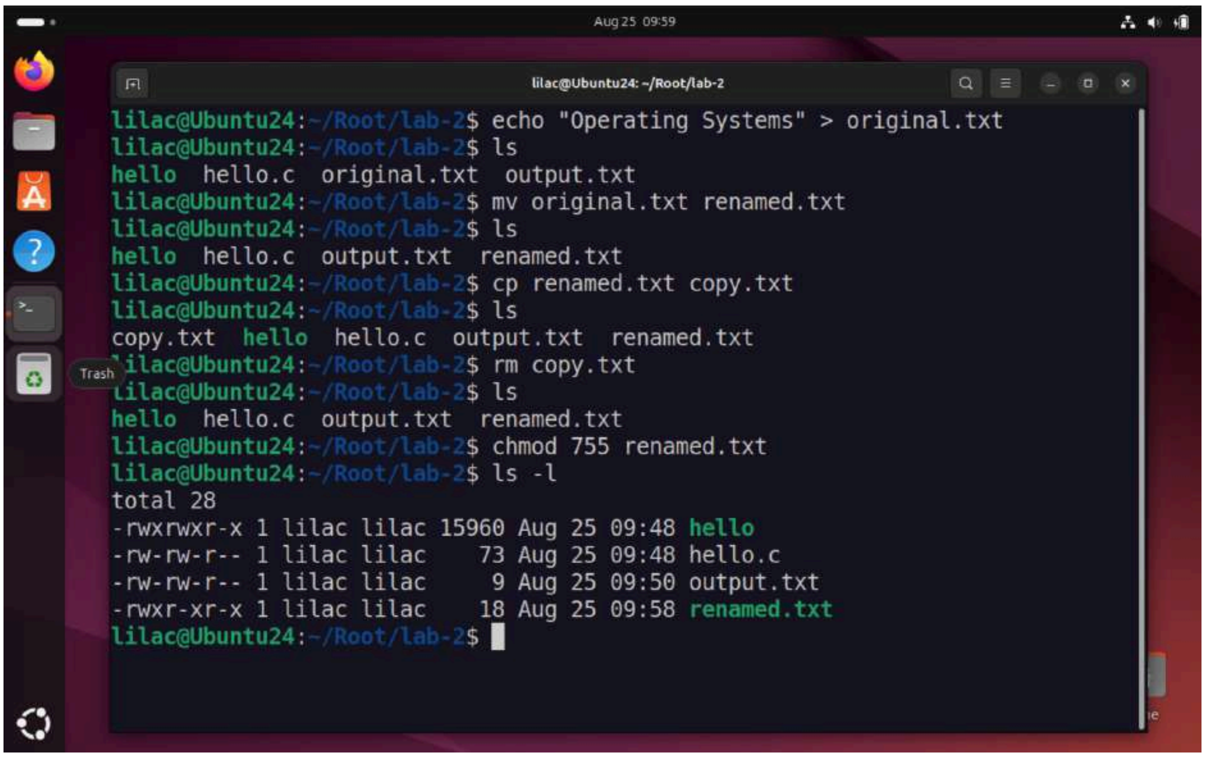


```
lilac@Ubuntu24:~/Root/lab-2$ touch hello.c
lilac@Ubuntu24:~/Root/lab-2$ nano hello.c
lilac@Ubuntu24:~/Root/lab-2$ gcc -o hello hello.c
lilac@Ubuntu24:~/Root/lab-2$ ls
hello  hello.c
lilac@Ubuntu24:~/Root/lab-2$ touch output.txt
lilac@Ubuntu24:~/Root/lab-2$ ./hello >> output.txt
```



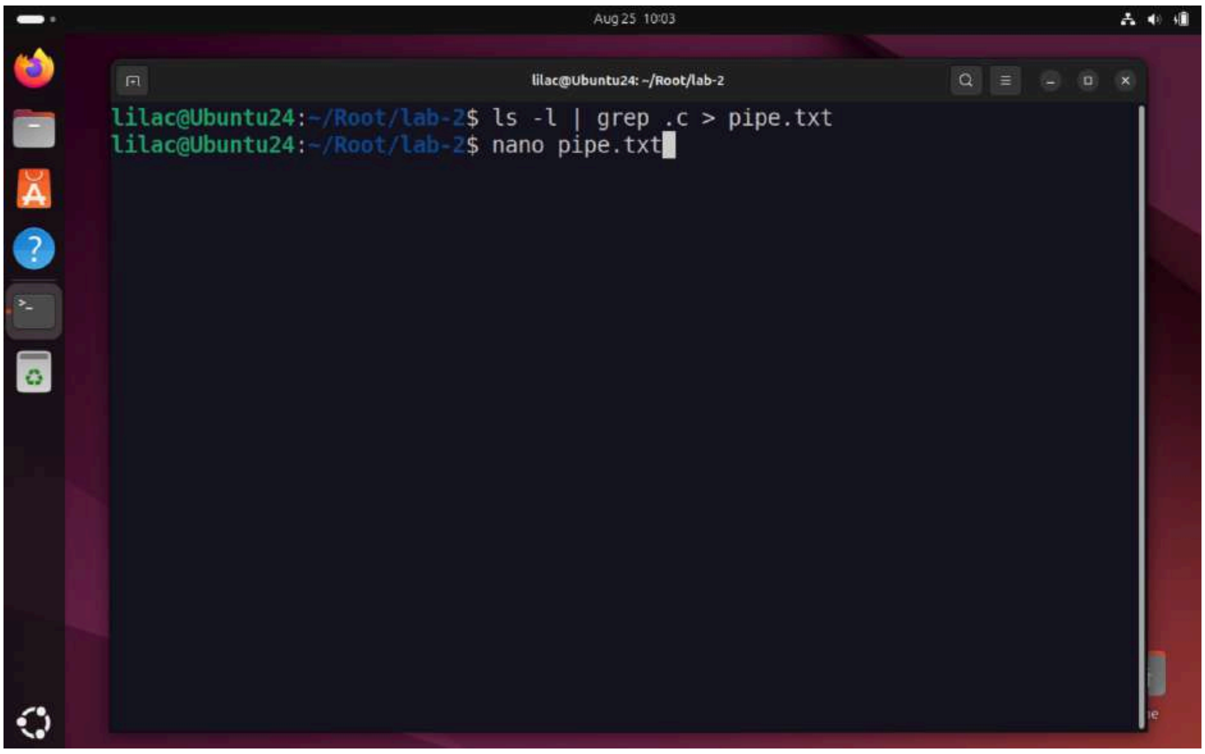
```
  GNU nano 7.2                    output.txt
Hello OS



                       [ Read 1 line ]
```

3.

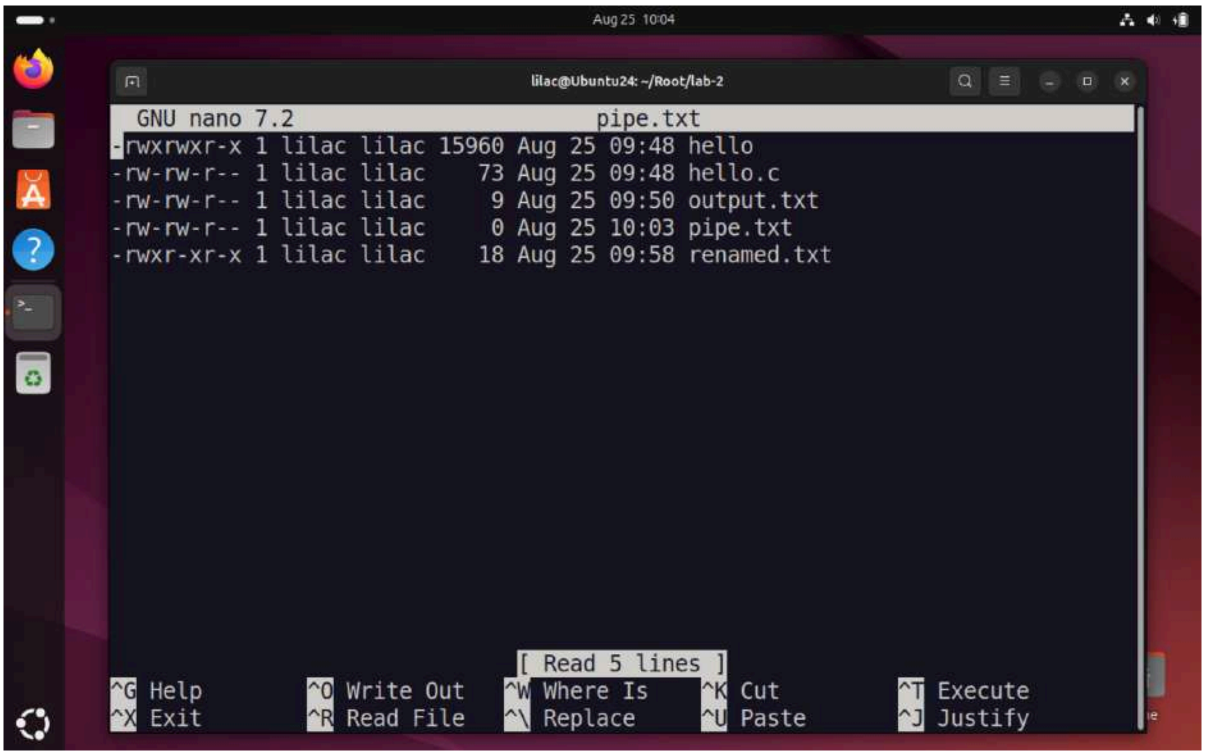


```
lilac@Ubuntu24:~/Root/lab-2$ echo "Operating Systems" > original.txt
lilac@Ubuntu24:~/Root/lab-2$ ls
hello  hello.c  original.txt  output.txt
lilac@Ubuntu24:~/Root/lab-2$ mv original.txt renamed.txt
lilac@Ubuntu24:~/Root/lab-2$ ls
hello  hello.c  output.txt  renamed.txt
lilac@Ubuntu24:~/Root/lab-2$ cp renamed.txt copy.txt
lilac@Ubuntu24:~/Root/lab-2$ ls
copy.txt  hello  hello.c  output.txt  renamed.txt
lilac@Ubuntu24:~/Root/lab-2$ rm copy.txt
lilac@Ubuntu24:~/Root/lab-2$ ls
hello  hello.c  output.txt  renamed.txt
lilac@Ubuntu24:~/Root/lab-2$ chmod 755 renamed.txt
lilac@Ubuntu24:~/Root/lab-2$ ls -l
total 28
-rwxrwxr-x 1 lilac lilac 15960 Aug 25 09:48 hello
-rw-rw-r-- 1 lilac lilac    73 Aug 25 09:48 hello.c
-rw-rw-r-- 1 lilac lilac     9 Aug 25 09:50 output.txt
-rwxr-xr-x 1 lilac lilac    18 Aug 25 09:58 renamed.txt
lilac@Ubuntu24:~/Root/lab-2$ 
```

4.

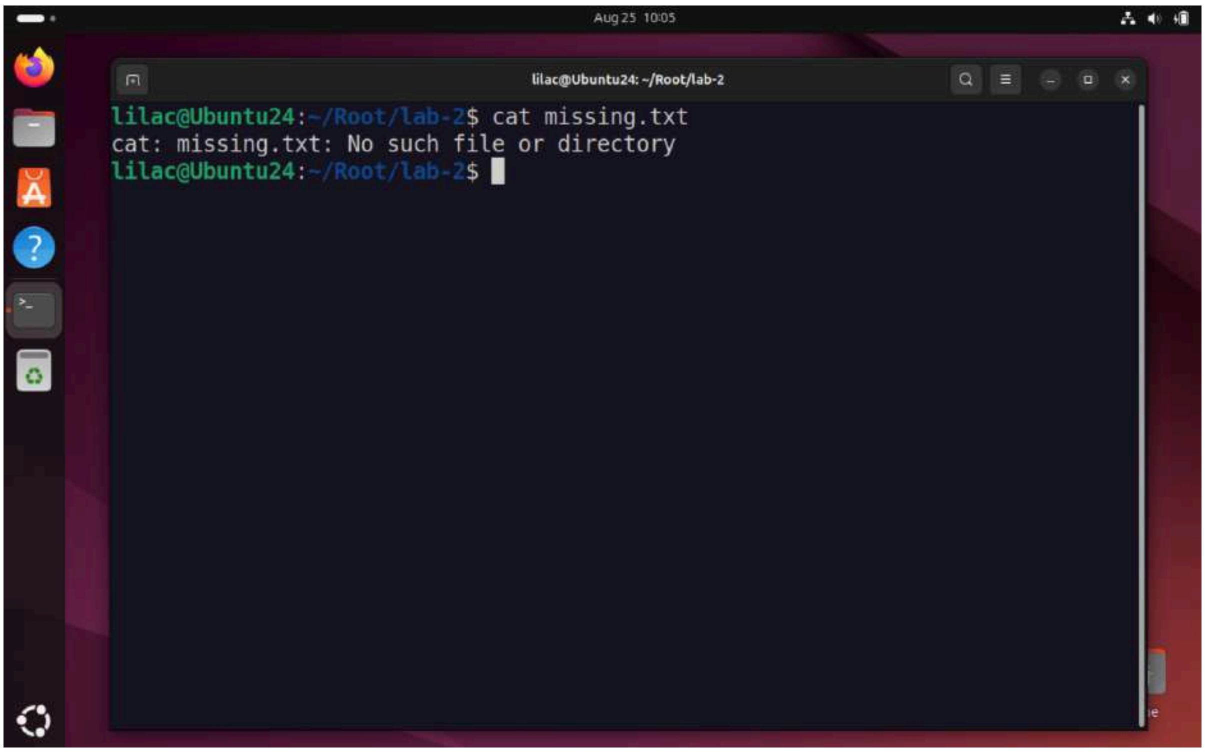


```
lilac@Ubuntu24:~/Root/lab-2$ ls -l | grep .c > pipe.txt
lilac@Ubuntu24:~/Root/lab-2$ nano pipe.txt
```



```
  GNU nano 7.2                                   pipe.txt
-rwxrwxr-x 1 lilac lilac 15960 Aug 25 09:48 hello
-rw-rw-r-- 1 lilac lilac    73 Aug 25 09:48 hello.c
-rw-rw-r-- 1 lilac lilac     9 Aug 25 09:50 output.txt
-rw-rw-r-- 1 lilac lilac     0 Aug 25 10:03 pipe.txt
-rwxr-xr-x 1 lilac lilac    18 Aug 25 09:58 renamed.txt

                              [ Read 5 lines ]
^G Help          ^O Write Out     ^W Where Is      ^K Cut           ^T Execute
^X Exit          ^R Read File     ^\ Replace       ^U Paste         ^J Justify
```

5.



```
lilac@Ubuntu24:~/Root/lab-2$ cat missing.txt
cat: missing.txt: No such file or directory
lilac@Ubuntu24:~/Root/lab-2$ 
```

The OS(Ubuntu) reported this error because missing.txt does not exist in the current working directory. The "cat" command makes a system call to open the file which triggers the open() system call. The kernel checks the current working directory of the process but the file doesn't exist. Therefore the lookup fails. The system call, open(), returns '-1' indicating an error occurred, and the 'cat' handles that error by returning an error message: "cat: missing.txt: No such file or directory".

6.

Command Used	**OS Service Demonstrated**	**Short Explanation**

$ touch hello.c
$ nano hello.c
$ gcc -o hello hello.c
	File Systems.
	Program Execution.
	Resource Allocation.
	I/O Operations.
		Create the hello.c file. Use nano, a command-line text editor to add code to print "Hello OS".
		Compile using gcc and execute the program.
		OS allocates memory for the program
		printf() calls write() under the hood.

$ ./hello >> output.txt	Program Execution Resource Management	Execute program. OS handles the process management and allocates the necessary resources (memory).

	File Systems I/O Operations	Redirects the output of the program to a file (open() the file and write() to it)

$ echo "Operating Systems" > original.txt
$ mv original.txt renamed.txt
$ cp renamed.txt copy.txt
$ rm copy.txt
$ chmod 755 renamed.txt
	I/O Operations	'echo' uses 'write()' to display on the terminal. But since '>' is used, it redirects the output to a different file (original.txt)

	File Systems	Rename the file using 'mv' command. Duplicate the file using 'cp' command. Delete the file using the 'rm' command. And change the permissions of the file using 'chmod 755'.
$ ls -l | grep .c > pipe.txt $ nano pipe.txt	Program Execution	The OS launches 'ls' and 'grep' as processes.
	Communication	The '|' command allows the 'ls' and 'grep' command to share data
	I/O Operations	This collected data is then outputted to a 'pipe.txt' using write()
	File Systems	'nano' command uses the open() command to open 'pipe.txt'
	Security and Protection	The command 'ls -l' lists the file and their permissions. Also, simply executing a program and commands, like 'grep' requires the OS to check if the user has the write permissions to execute it.
$ cat missing.txt	Program Execution	Using the 'cat' command makes the OS run the 'cat' program.
	File Systems Security and Protection	The 'cat' command attempts to open a file using open(), and the OS checks for its existence. The OS also checks if I have the right permissions to open the program
	Error Detection	The OS detects that the file does not exist and returns an error code.

7. Reflection

The OS services act as an abstraction layer to the more complex layer of the hardware for us users and developers. It gives us functionality we can use to operate with the hardware such as file operations, process control, and connecting external hardware(I/O devices) to our system. It also handles a lot of the complex tasks like allocating resources for the programs, handling errors, and securing our devices. All of this allow us to easily utilize or device, all thanks to the operating system.