Operating Systems

Lab #01
Title: Read System Call

Compiled By: Alina Liaquat
Roll-No: BSCSM-A-23-19
BSCS-MORNING-4th(A)
Submitted to: Sir Umar Malik

UNIVERSITY OF LAYYAH MAIN CAMPUS HAFIZABAD

1.1 Objectives

1. To understand what a system call is and how user programs interact with the operating system
2. To learn how the read(), open(), write(), and close() system calls work in Linux
3. To observe how file descriptors are created and used by the OS
4. To practice using the nano editor and basic terminal commands
5. To analyze the system call flow using a simple C program

1.2 Concept

A **system call** is an interface that allows a user program to request a service from the operating system.

User programs cannot directly access hardware or kernel memory, so they use system calls to perform tasks

such as file handling, process creation, memory allocation, and input/output operations.

The **read()** system call is used to transfer data from a file into a buffer in memory.

To read a file, it must first be opened using **open(),** which returns a file descriptor.

The **write()** system call sends data to an output stream (e.g. screen or file).

The **close()** system call is used to release the file descriptor after operations are completed. This lab introduces how low-level file operations work behind high-level functions.

1: READ SYSTEM CALL

```
alinagame@DESKTOP-4TJUFLT: ~
alinagame@DESKTOP-4TJUFLT:~$ mkdir -p ~/os_lab
alinagame@DESKTOP-4TJUFLT:~$ ls
game_project  os_lab  projects
```

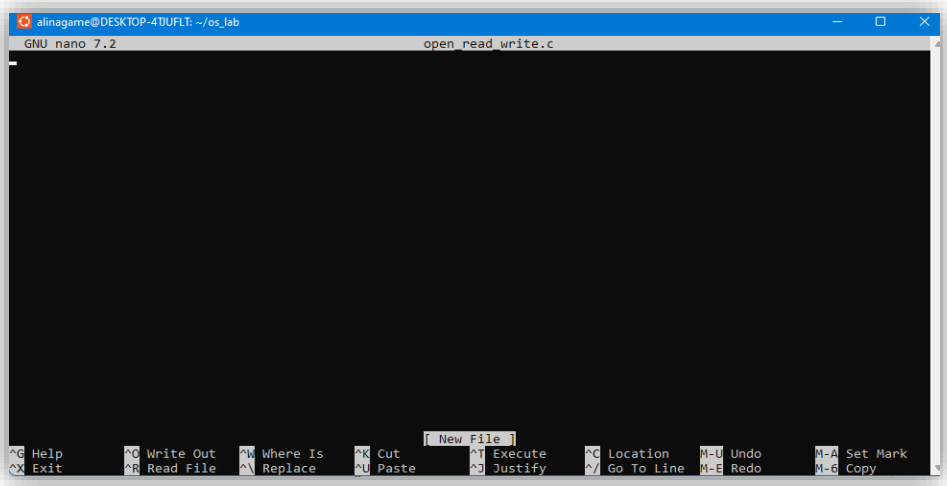
Explanation:

1. mkdir = make directory
2. -p = makes sure error would not occur even if directory already exists
3. ~/ = home folder
4. os_lab = folder name

```
alinagame@DESKTOP-4TJUFLT:~$ cd ~/os_lab
alinagame@DESKTOP-4TJUFLT:~/os_lab$
alinagame@DESKTOP-4TJUFLT:~/os_lab$ nano open_read_write.c
```

Opens nano , terminal text editor where we can read/write/copy/paste the programs.

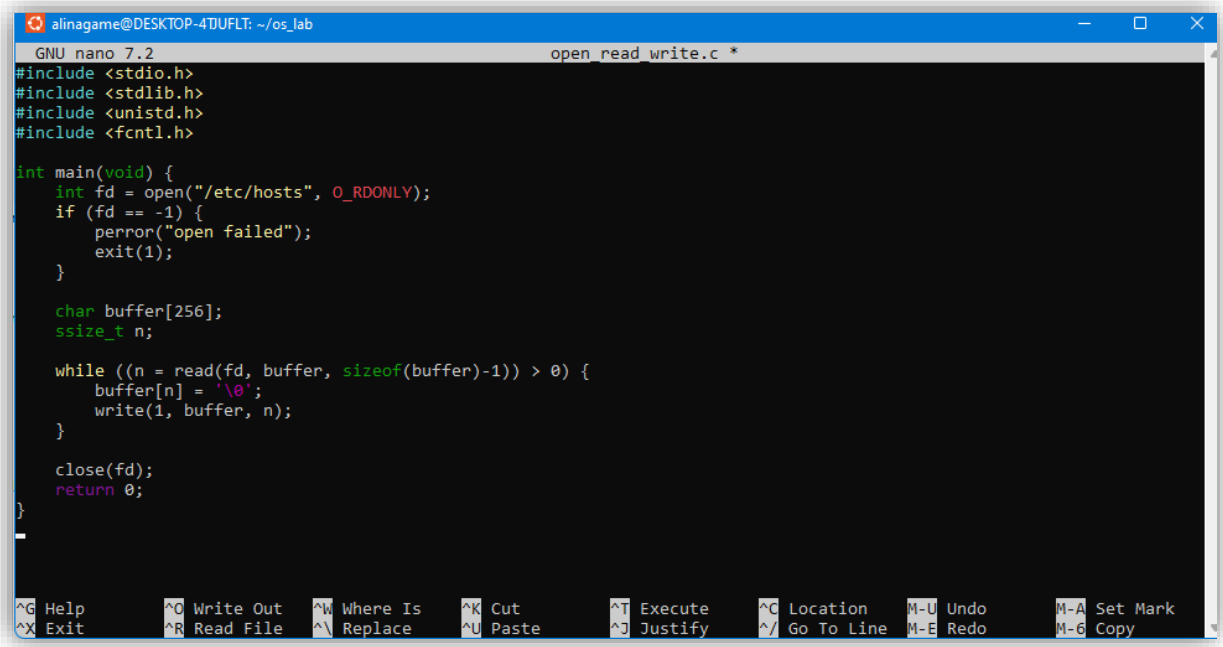
After running above command following interface opens



Nano Shortcuts:

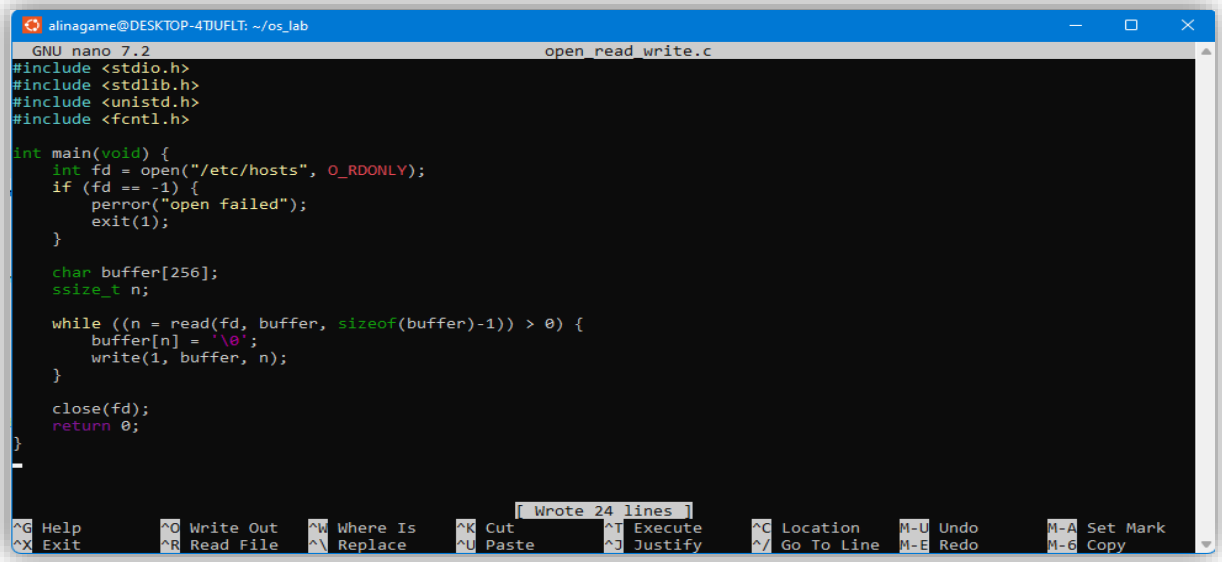
Key	Meaning
Ctrl + O	save (Write Out)
Enter	confirm file name
Ctrl + X	exit

Copy/Paste Code



Ctrl + O → Enter

Exit: **Ctrl + X**



Line	Meaning
open("/etc/hosts", O_RDONLY)	OS request: file open
fd	file descriptor: OS Number which represents file
read()	OS request: bring file data to buffer

write(1, ...)	1 means screen
close()	file close

```
alinagame@DESKTOP-4TJUFLT:~/os_lab$ gcc open_read_write.c -o read_prog
```

We use gcc (GNU C compiler)

There was no error it means code is error free

Run Program:

```
alinagame@DESKTOP-4TJUFLT:~/os_lab$ ./read_prog
# This file was automatically generated by WSL. To stop automatic generation of this file, add the following entry to /etc/wsl.conf:
# [network]
# generateHosts = false
127.0.0.1       localhost
127.0.1.1       DESKTOP-4TJUFLT.localdomain      DESKTOP-4TJUFLT
192.168.43.247  host.docker.internal
192.168.43.247  gateway.docker.internal
127.0.0.1       kubernetes.docker.internal

# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

2: WRITE SYSTEM CALL

Open nano:

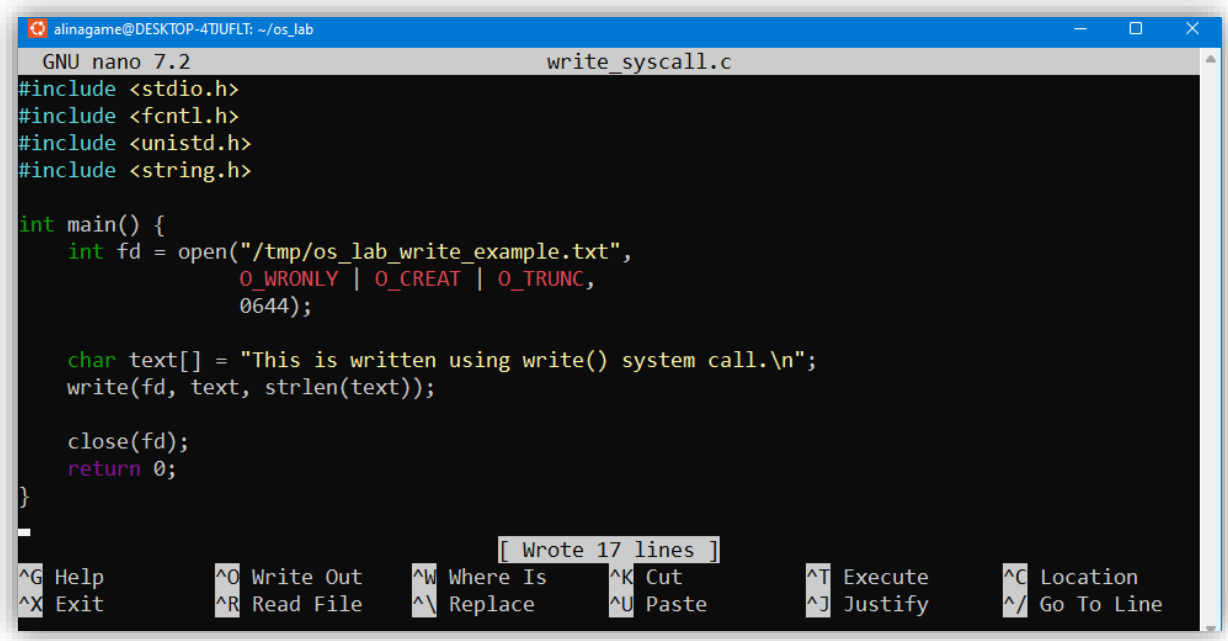
```
alinagame@DESKTOP-4TJUFLT:~/os_lab$ nano write_syscall.c
```

Paste Code

```
alinagame@DESKTOP-4TJUFLT: ~/os_lab
  GNU nano 7.2                                write_syscall.c *
#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>
#include <string.h>

int main() {
    int fd = open("/tmp/os_lab_write_example.txt",
                  O_WRONLY | O_CREAT | O_TRUNC,
                  0644);

    char text[] = "This is written using write() system call.\n";
    write(fd, text, strlen(text));

    close(fd);
    return 0;
}

^G Help        ^O Write Out   ^W Where Is    ^K Cut         ^T Execute     ^C Location
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify     ^/ Go To Line
```

Save + Exit

Ctrl+O → Enter → Ctrl+X



Compile & Run:

```
alinagame@DESKTOP-4TJUFLT:~/os_lab$ gcc write_syscall.c -o write_prog
```

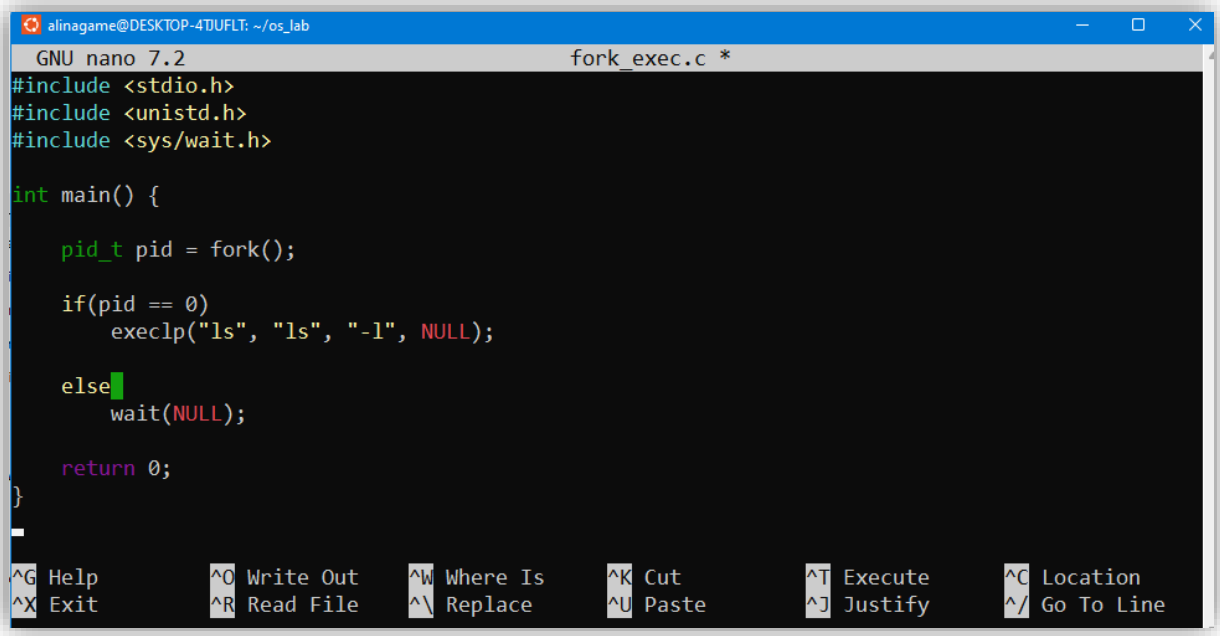
Verify file was created:

```
alinagame@DESKTOP-4TJUFLT:~/os_lab$ cat /tmp/os_lab_write_example.txt
This is written using write() system call.
alinagame@DESKTOP-4TJUFLT:~/os_lab$
```

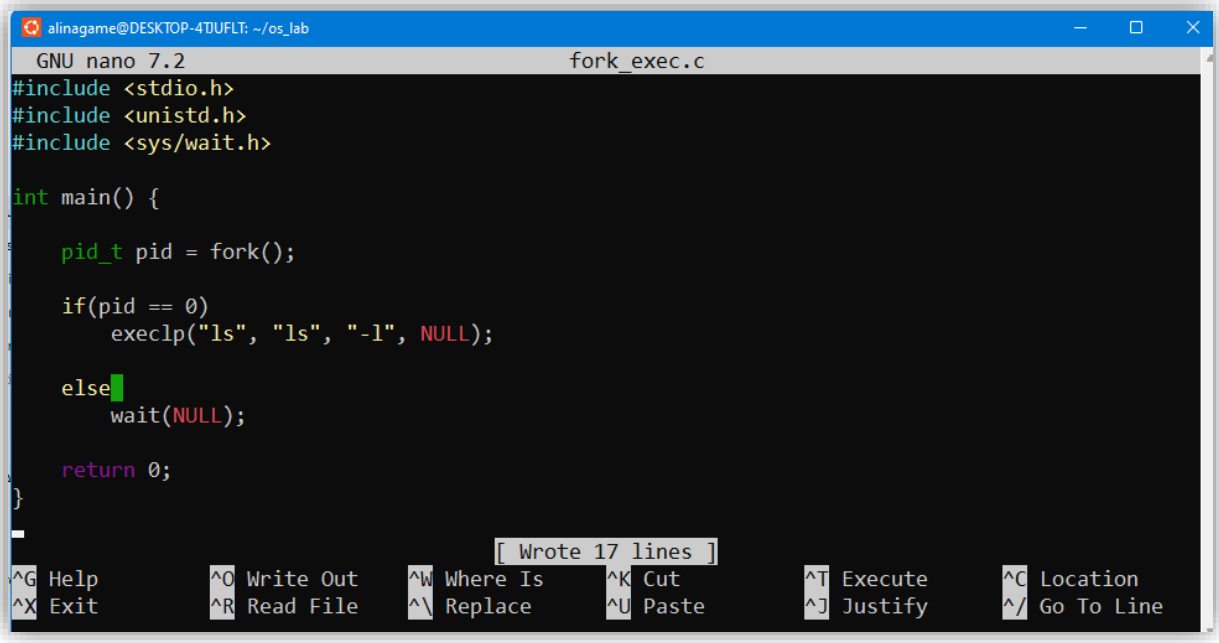
3: Fork + Exec

```
alinagame@DESKTOP-4TJUFLT:~/os_lab$ nano fork_exec.c
```

Paste Code



Save & Exit the code



Compile & Run

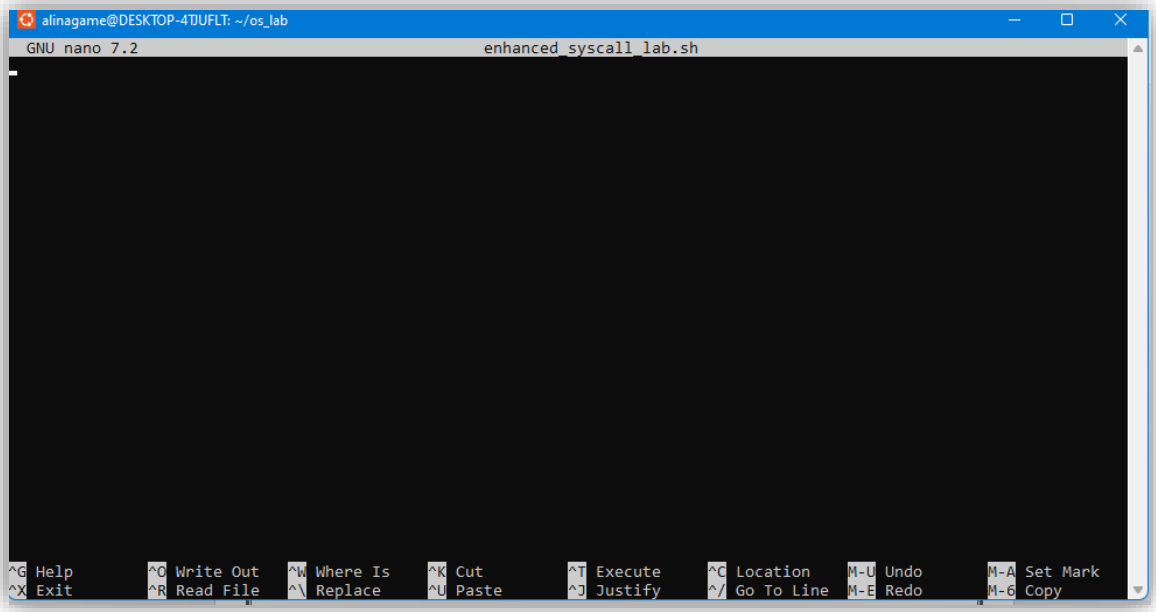
```
alinagame@DESKTOP-4TJUFLT:~/os_lab$ ./fork_exec_prog
total 60
-rw-r--r-- 1 alinagame alinagame   207 Nov 17 20:00 fork_exec.c
-rwxr-xr-x 1 alinagame alinagame 16048 Nov 17 20:00 fork_exec_prog
-rw-r--r-- 1 alinagame alinagame   405 Nov 17 19:44 open_read_write.c
-rwxr-xr-x 1 alinagame alinagame 16232 Nov 17 19:46 read_prog
-rwxr-xr-x 1 alinagame alinagame 16144 Nov 17 19:54 write_prog
-rw-r--r-- 1 alinagame alinagame   351 Nov 17 19:50 write_syscall.c
```

Strace traces system level calls

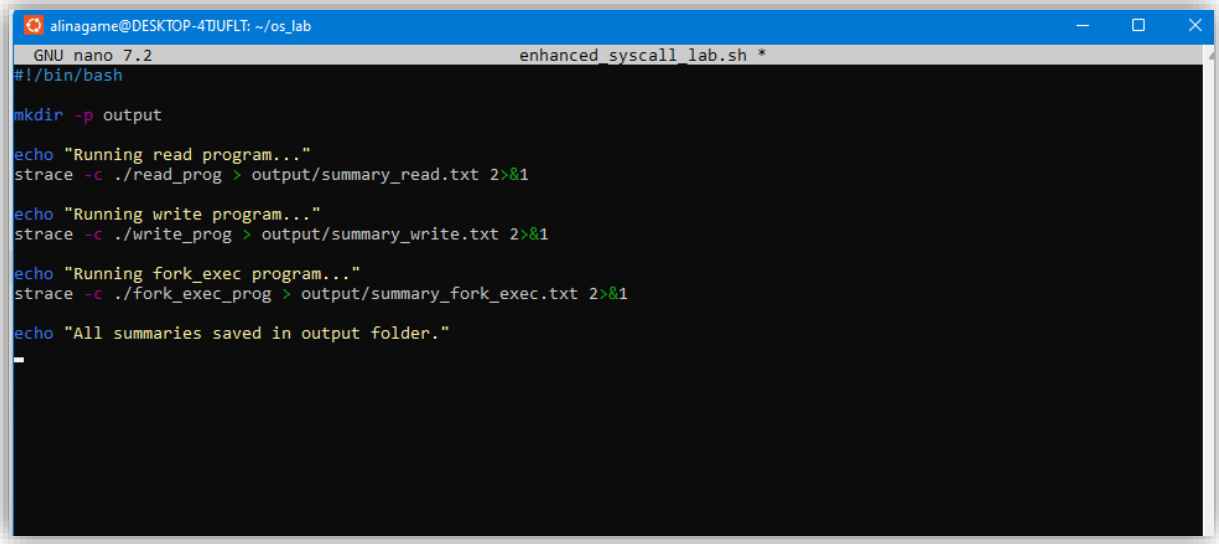
```
alinagame@DESKTOP-4TJUFLT:~/os_lab$ strace -c ./read_prog
# This file was automatically generated by WSL. To stop automatic generation of this file, add the following entry to /etc/wsl.conf:
# [network]
# generateHosts = false
127.0.0.1       localhost
127.0.1.1       DESKTOP-4TJUFLT.localdomain      DESKTOP-4TJUFLT
192.168.43.247  host.docker.internal
192.168.43.247  gateway.docker.internal
127.0.0.1       kubernetes.docker.internal

# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
% time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- --------- ----------------
 24.27    0.000952         119         8           mmap
 13.49    0.000529         176         3           openat
  8.98    0.000352         117         3           write
  8.77    0.000344          68         5           read
  7.78    0.000305         152         2           fstat
```

Open nano

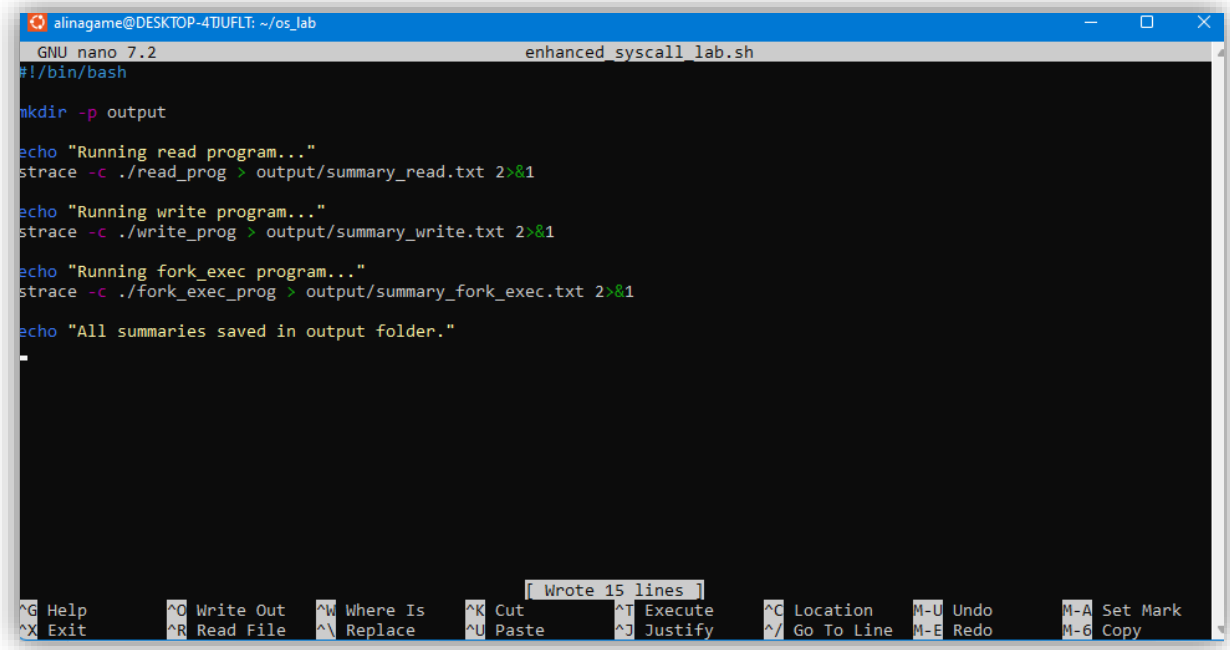
```
alinagame@DESKTOP-4TJUFLT:~/os_lab$ nano enhanced_syscall_lab.sh
```



Create Script



Pasting the commands



Making it executable

```
alinagame@DESKTOP-4TJUFLT:~/os_lab$ chmod +x enhanced_syscall_lab.sh
```

Run

```
alinagame@DESKTOP-4TJUFLT:~/os_lab$ ./enhanced_syscall_lab.sh
Running read program...
Running write program...
Running fork_exec program...
All summaries saved in output folder.
```

CONCLUSION:

In this lab, I've learned how the write() system call works and how files are created and modified at the kernel level. I observed how system calls differ from normal library functions and understood how Linux handles low-level file output.