Date: _____

Batch: E2 **Roll No.: 16010123325**

Experiment No. 03

Grade: AA / AB / BB / BC / CC / CD /DD

Signature of the Staff In-charge with date

TITLE: System calls

AIM: To understand the working Process based system calls.

Expected Outcome of Experiment:

CO 1. To introduce basic concepts and functions of operating systems.

Books/ Journals/ Websites referred:

1. **Silberschatz A., Galvin P., Gagne G. "Operating Systems Principles", Willey Eight edition.**
2. **William Stallings "Operating Systems" Person, Seventh Edition Edition.**
3. **Sumitabha Das " UNIX Concepts & Applications", McGraw Hill Second Edition.**

Pre Lab/ Prior Concepts:

System Calls Provide the Interface between a process and the OS.

System calls are usually made when a process in user mode requires access to a resource.

Then it requests the kernel to provide the resource via a system call.

System calls are required in the following situations −

1) If a file system requires the creation or deletion of files.

2) Reading and writing from files also require a system call.

3) Creation and management of new processes.

4) Network connections also require system calls. This includes sending and receiving packets.

5) Access to a hardware devices such as a printer, scanner etc. requires a system call.

Description of the application to be implemented:

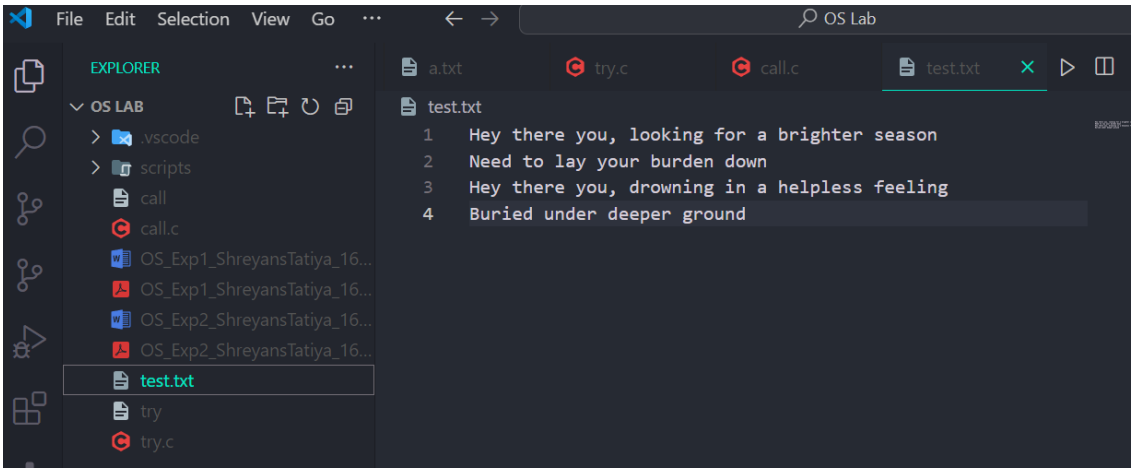
Program for System Call:

1. Write a Program for creating process using System call (E.g fork()). Create a child process. Display the details about that process using getpid and getppid functions in each block. Also print the return value of fork() system call in each block.
   In the child process, Open a text file using file system calls and read the contents of the text file and display it.

Implementation details: (Screen shots)

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

int main() {
    int f = fork();
    if (f < 0) {
        printf("Fork failed\n");
    }
    else if (f == 0) {
        printf("Child process: %d\n", getpid());
        printf("Parent process: %d\n", getppid());
        printf("Fork return value in child: %d\n", f);

        int fd = open("test.txt", O_RDONLY);
        if (fd < 0) {
            printf("Error opening file\n");
            return 1;
        }

        char buffer[1024];
        int bytes_read;

        printf("\nReading file contents:\n");
        printf("----------------------\n");
        while ((bytes_read = read(fd, buffer, sizeof(buffer))) > 0) {
            write(STDOUT_FILENO, buffer, bytes_read);
        }

        close(fd);
    }
```

```
    else {
        printf("Parent process: %d\n", getpid());
        printf("Child process: %d\n", f);
        printf("Fork return value in parent: %d\n", f);
    }
    return 0;
}
```




```
root@Shrey:/mnt/c/Users/Shrey/OneDrive/Desktop/KJSCE/SEM-4/OS Lab#
gcc call.c -o call
root@Shrey:/mnt/c/Users/Shrey/OneDrive/Desktop/KJSCE/SEM-4/OS Lab#
./call
Parent process: 405
Child process: 406
Fork return value in parent: 406
Child process: 406
Parent process: 405
Fork return value in child: 0
root@Shrey:/mnt/c/Users/Shrey/OneDrive/Desktop/KJSCE/SEM-4/OS Lab#

Reading file contents:
--------------------
Hey there you, looking for a brighter season
Need to lay your burden down
Hey there you, drowning in a helpless feeling
Buried under deeper ground
```

Conclusion :

The program demonstrates the use of system calls like fork(), getpid, and getppid to create and manage processes, alongside file operations to access resources. It highlights how system calls enable user-mode processes to interact with the OS kernel, showcasing process hierarchy and controlled resource handling in Linux.

Post Lab Descriptive Questions

1) Describe System Call Interface.

Ans:

System Call Interface:

- Acts as a bridge between user applications and the OS kernel.
- Provides a controlled API (e.g., fork(), open(), read()) to access kernel services.
- Triggers a switch from **user mode** to **kernel mode** for secure resource access.
- Each system call is identified by a unique number and invoked via libraries (e.g., glibc).

2) List the types of System Calls.

- **Process Control**: Create/terminate processes (e.g., fork(), exec(), exit()).
- **File Management**: File operations (e.g., open(), read(), write()).
- **Device Management**: Hardware access (e.g., ioctl(), read(), write()).
- **Information Maintenance**: Get system data (e.g., getpid(), time()).
- **Communication**: Inter-process communication (e.g., pipe(), shmget()).

Date: _____ **Signature of faculty in-char**