**Batch: C3 Roll No.:16010123217**

**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:**

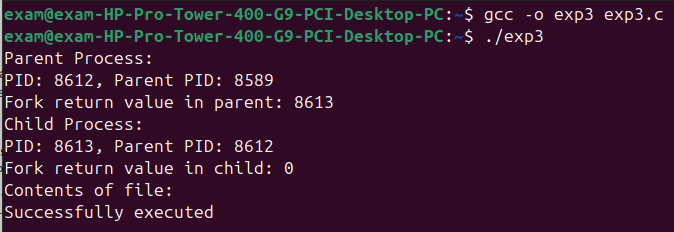
**Program for System Call:**

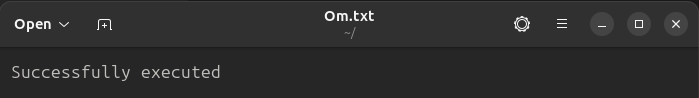
1. Write a Program for creating process using System call (E.g fork()). Create a child process. Display the details about that process using getpid and getppid functions in each block. Also print the return value of fork() system call in each block.

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)

Code:  
#include <stdio.h>  
#include <stdlib.h>  
#include <sys/types.h>  
#include <sys/wait.h>  
#include <unistd.h>  
#include <fcntl.h>  
  
int main() {  
    pid\_t retval;  
  
    retval = fork();  
  
    if (retval < 0) {  
        perror("Fork failed");  
        exit(1);  
    } else if (retval == 0) {  
        printf("Child Process:\n");  
        printf("PID: %d, Parent PID: %d\n", getpid(), getppid());  
        printf("Fork return value in child: %d\n", retval);  
  
         
        int file = open("Om.txt", O\_RDONLY);  
        if (file < 0) {  
            perror("File open failed");  
            exit(1);  
        }  
  
        char buffer[256];  
        ssize\_t bytesRead = read(file, buffer, sizeof(buffer) - 1);  
        if (bytesRead < 0) {  
            perror("File read failed");  
            exit(1);  
        }  
  
        buffer[bytesRead] = '\0';  
        printf("Contents of file:\n%s\n", buffer);  
        close(file);  
    } else {  
        printf("Parent Process:\n");  
        printf("PID: %d, Parent PID: %d\n", getpid(), getppid());  
        printf("Fork return value in parent: %d\n", retval);  
  
        wait(NULL);  
    }  
  
    return 0;  
}

Screenshots:  


Om.txt  


**Conclusion :**

From this experiment, we learnt the working of fork and displaying the contents of the file

**Post Lab Descriptive Questions**

1. Describe System Call Interface.

Ans. A system call is an interface between user programs and the OS kernel, allowing applications to request services like file operations, process control, and hardware access.

Purpose of System Calls

User-Kernel Boundary – Prevents direct access to system resources.

Resource Management – Manages CPU, memory, and storage.

Streamlined Development – Abstracts hardware complexities.

Security & Access Control – Ensures permission checks.

Inter-Process Communication (IPC) – Enables data exchange via pipes, message queues, and shared memory.

Network Operations – Supports communication over networks.

How System Calls Work

Request – Application invokes a system call (e.g., read()).

Context Switch – Transitions from user mode to kernel mode.

Identification – Kernel identifies the system call.

Execution – Kernel executes the requested function.

Return Values – Kernel prepares results for the application.

Switch Back – Context switches to user mode.

Resume – Application continues execution with system call results.

1. List the types of System Calls.

Ans.

Process Control – Create, terminate, execute, schedule, and wait for processes.

File Management – Read, write, open, close, modify, and delete files.

Device Management – Request access, set attributes, read/write, and map devices.

Information Maintenance – Retrieve/modify system attributes, time, and performance data.

Communication – Send/receive messages, sync processes, use shared memory, and network via sockets.

Security & Access Control – Manage resource access, permissions, and authentication.

**Date: 06/02/25 Signature of faculty in-charge**