**Batch: A-3 Roll No.: 16010122104**

**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.**

1. **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 a child process, Open the file using file system calls and read the contents and display.

**Implementation details:** (printout of code / screen shot)

import os

def child\_process():

    # Display child process information

    print(f"Child Process ID: {os.getpid()}")

    print(f"Parent Process ID (from Child): {os.getppid()}")

    # Open and read a file using file system calls

    try:

        with open("example.txt", "r") as file:

            content = file.read()

            print("\nFile content read by child process:\n")

            print(content)

    except FileNotFoundError:

        print("File 'example.txt' not found. Please ensure the file exists.")

def parent\_process():

    # Display parent process information

    print(f"Parent Process ID: {os.getpid()}")

# Main function

def main():

    # Fork to create a new process

    pid = os.fork()

    if pid > 0:

        # Parent process

        parent\_process()

    elif pid == 0:

        # Child process

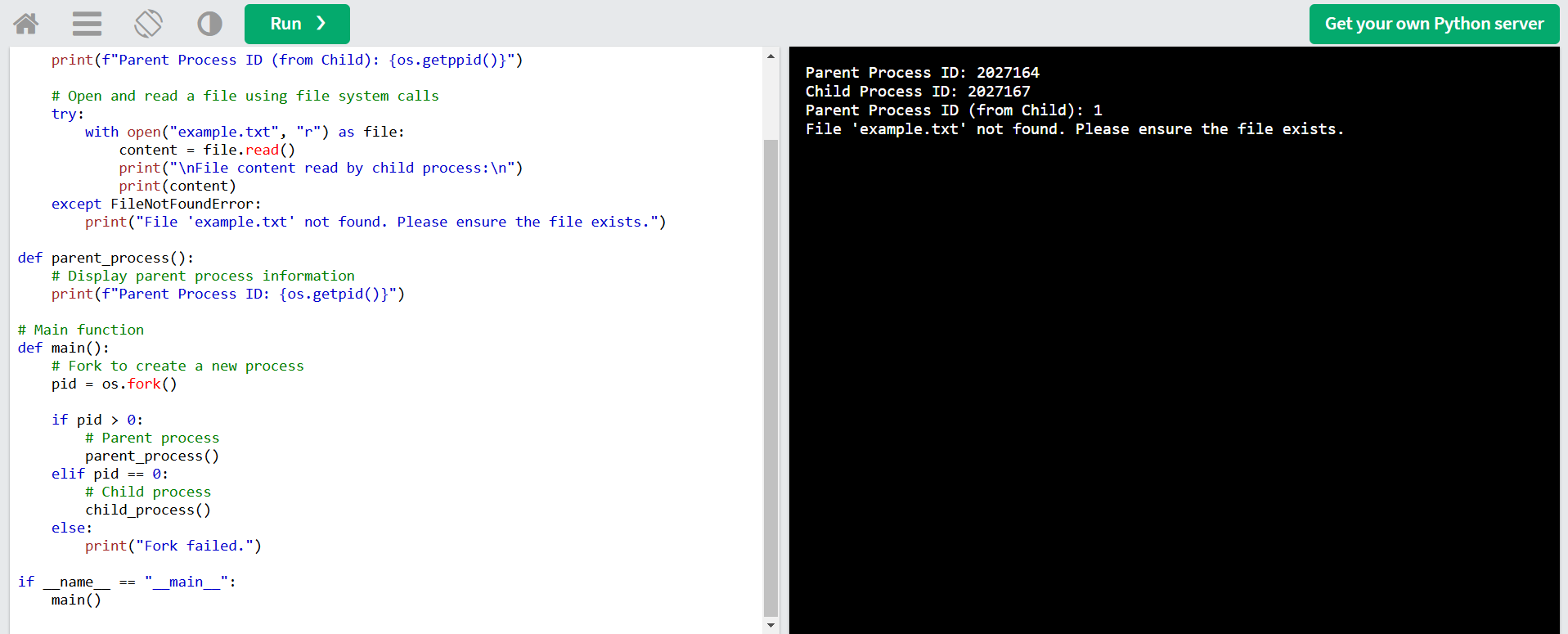
        child\_process()

    else:

        print("Fork failed.")

if \_\_name\_\_ == "\_\_main\_\_":

    main()

****

**Conclusion :**

We implemented systems calls using different programs.

**Post Lab Descriptive Questions**

1. Describe System Call Interface.

Ans:

The **System Call Interface (SCI)** is a crucial component of the operating system that provides a communication layer between user applications and the OS kernel. When applications need to perform low-level operations (like accessing hardware, creating processes, managing files, etc.), they cannot directly access the kernel. Instead, they make system calls, which are special functions that request services from the kernel.

When an application calls a system function, the SCI translates this request into a system call and switches the context from user mode to kernel mode, allowing controlled access to the kernel’s privileged resources. This interface ensures that applications have safe and limited access to the system’s core functionalities while maintaining security and stability.

1. List the types of System Calls.

Ans:

System calls can be broadly classified into several categories:

* **Process Control**: These system calls manage processes, including their creation, termination, and synchronization.
  + Examples: fork(), exec(), wait(), exit(), kill()
* **File Management**: These calls handle files and directories, including creating, reading, writing, and deleting files.
  + Examples: open(), read(), write(), close(), unlink()
* **Device Management**: These calls are used to request access to hardware devices, send data to them, or receive data from them.
  + Examples: ioctl(), read(), write(), close()
* **Information Maintenance**: System calls that provide information about and manage system resources and settings.
  + Examples: getpid(), alarm(), sleep(), setpriority()
* **Communication**: These calls facilitate communication between processes, often over network connections or shared memory.
  + Examples: pipe(), shmget(), mmap(), send(), recv()

**Date: 25/10/2024 Signature of faculty in-charge**