**Batch: D - 1 Roll No.: 16010122096**

**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:**

import os

def main():

filename = 'file.txt'

try:

fd = os.open(filename, os.O\_RDWR | os.O\_CREAT | os.O\_TRUNC)

except OSError as e:

print(f"Error opening file: {e}")

return

try:

os.write(fd, "Message from parent process.".encode('utf-8'))

except OSError as e:

print(f"Error writing to file: {e}")

pid = os.fork()

if pid == 0:

child\_pid = os.getpid()

parent\_pid = os.getppid()

print(f"Child Process ID: {child\_pid}")

print(f"Parent Process ID: {parent\_pid}")

try:

os.lseek(fd, 0, os.SEEK\_SET)

content = os.read(fd, 1024).decode('utf-8')

print("Child read:")

print(content)

except OSError as e:

print(f"Error reading file: {e}")

try:

os.lseek(fd, 0, os.SEEK\_END)

os.write(fd, "\nMessage from child process.".encode('utf-8'))

except OSError as e:

print(f"Error writing to file: {e}")

os.\_exit(0)

else:

os.wait()

try:

os.lseek(fd, 0, os.SEEK\_SET)

content = os.read(fd, 1024).decode('utf-8')

print("\nFinal contents of the file (from parent):")

print(content)

except OSError as e:

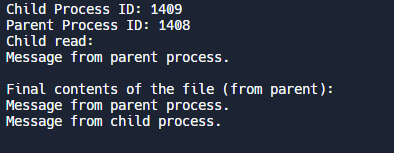
print(f"Error reading file: {e}")

os.close(fd)

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

main()

**Output:**



**Conclusion :**

This experiment demonstrates the use of process-based system calls, including `fork()` for creating child processes, and file operations for shared access. It highlights how processes interact and manage file offsets.

**Post Lab Descriptive Questions**

1. Describe System Call Interface.

The **System Call Interface** is a mechanism that allows user applications to request services from the operating system. It acts as a bridge between user space and kernel space, enabling applications to perform operations such as file manipulation and process management securely and efficiently.

1. List the types of System Calls.
   1. **Process Control**:

* fork(): Creates a new process.
* exec(): Replaces the current process image.
* exit(): Terminates a process.
* wait(): Waits for a process to change state.
  1. **File Management**:
* open(): Opens a file.
* read(): Reads from a file.
* write(): Writes to a file.
* close(): Closes a file.
  1. **Device Management**:
* ioctl(): Performs device-specific operations.
* read(): Reads from a device.
* write(): Writes to a device.
  1. **Information Maintenance**:
* getpid(): Gets the process ID.
* setuid(): Sets the user ID.
  1. **Inter-process Communication (IPC)**:
* pipe(): Creates a pipe for IPC.
* shmget(): Allocates shared memory.

**Date: 22 / 08 / 2024 Signature of faculty in-charge**