**Batch: C1 Roll No.:16010122221**

**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:**

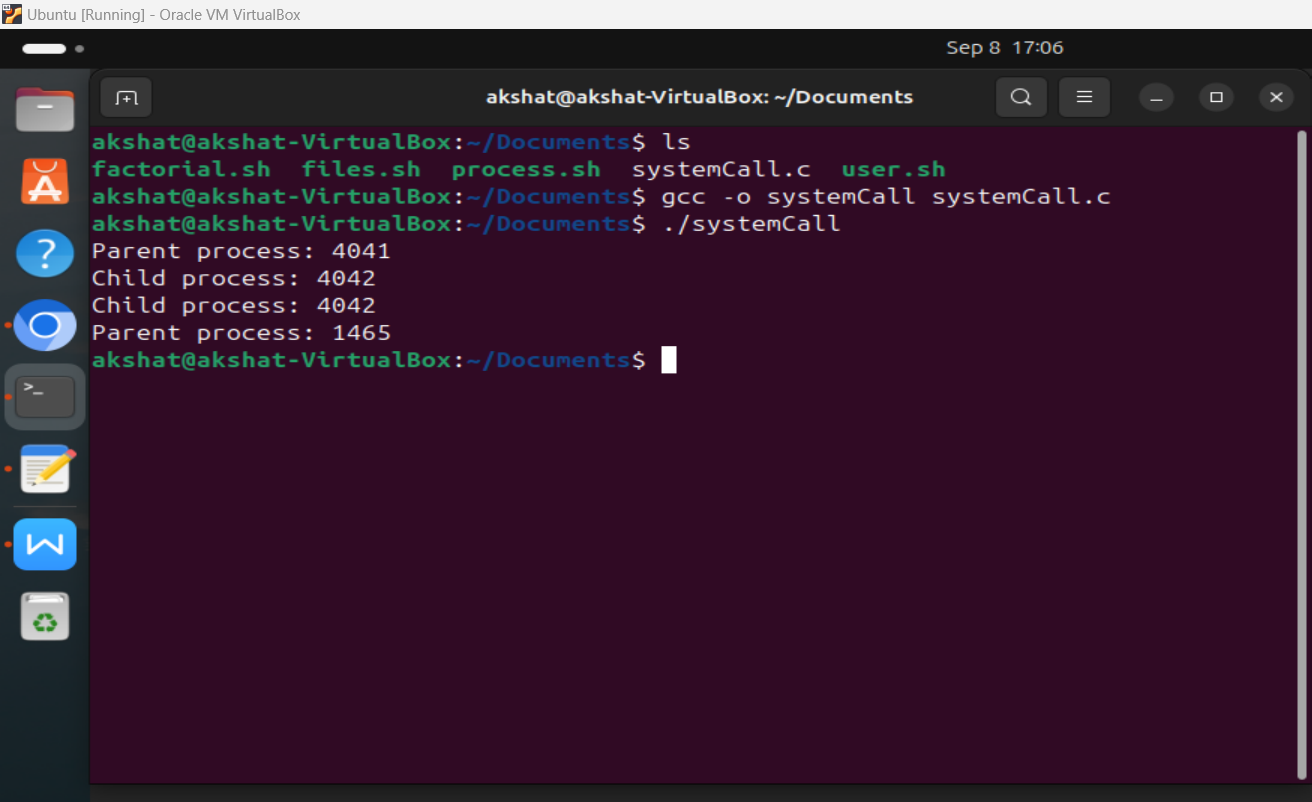
**Program for System Call:**

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)

**Program for system call code:-**





**Post Lab Descriptive Questions**

1. Describe System Call Interface.

The System Call Interface (SCI) is the mechanism that allows user-level programs to request services from the operating system’s kernel. Since user applications run in user mode with limited access to system resources for security, they rely on system calls to perform tasks like file management, process control, memory allocation, and device communication.

When a program makes a system call, it uses a predefined API, such as `open()`, `read()`, or `write()`. Each system call has a unique number, and the program passes this number along with necessary parameters to the kernel through a software interrupt or trap. The kernel, running in kernel mode, processes the request and returns control to the user program, often with a return value indicating success or failure.

System calls are categorized based on functionality, including process control (`fork()`), file operations (`open()`, `read()`), and inter-process communication (`pipe()`, `shmget()`).

The SCI ensures security by preventing direct access to critical resources and provides abstraction, allowing developers to interact with hardware and resources without needing to manage low-level operations. This interface is essential for managing hardware and system resources efficiently while ensuring system stability.

1. List the types of System Calls.

System calls are typically categorized into five main types based on their functionality. These categories provide different services to programs interacting with the operating system:

1. Process Control\*:

System calls that manage processes, allowing creation, termination, and control of processes.

- Examples: `fork()`, `exec()`, `exit()`, `wait()`, `getpid()`, `kill()`

### 2. \*\*File Management\*\*:

These system calls manage files and directories, allowing the creation, deletion, reading, writing, and manipulation of files.

- Examples: `open()`, `read()`, `write()`, `close()`, `unlink()`, `lseek()`, `stat()`

3. Device Management:

System calls that handle devices, enabling the interaction between user programs and hardware devices.

- Examples: `ioctl()`, `read()`, `write()`, `close()`

4. Information Maintenance:

These system calls provide mechanisms to transfer information between user programs and the operating system, including time management, setting system data, and obtaining process or system status.

- Examples: `getpid()`, `getuid()`, `alarm()`, `sleep()`, `gettimeofday()`

5. Communication:

System calls that manage communication between processes, both within the same system and across a network.

- Examples: `pipe()`, `shmget()`, `msgsnd()`, `msgrcv()`, `socket()`, `recv()`, `send()`

Each category of system calls provides critical functionality for application programs to interact with the system's underlying resources and services.

**Date: \_\_\_\_\_\_\_\_\_\_\_\_\_ Signature of faculty in-charge**