

| **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)

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/wait.h>

int main() {

pid\_t pid;

FILE \*file;

char buffer[1000];

pid = fork();

if (pid < 0) {

fprintf(stderr, "Fork unsuccessful!");

return -1;

} else if (pid == 0) {

printf("Child PID: %d\n", getpid());

printf("Parent PID: %d\n", getppid());

file = fopen("woohoo.txt", "r");

if (file == NULL) {

fprintf(stderr, "Unable to open file\n");

return -1;

}

while (fgets(buffer, sizeof(buffer), file) != NULL) {

printf("%s", buffer);

}

fclose(file);

} else {

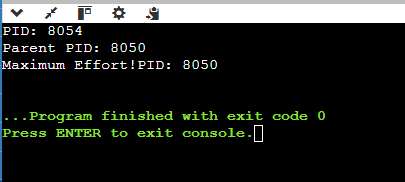
wait(NULL);

printf("Parent PID: %d\n", getpid());

}

return 0;

}



**Conclusion :**

**Post Lab Descriptive Questions**

1. Describe System Call Interface.

A **System Call Interface** is a crucial component of an operating system that provides the means for user programs to interact with the underlying hardware. It acts as an intermediary between user applications and the kernel, allowing these applications to request services or resources managed by the operating system. When an application needs to perform tasks like reading or writing files, creating processes, or communicating over a network, it makes system calls.

Key points about the System Call Interface:

* **Abstraction Layer**: It abstracts the complex details of hardware operations, providing a simpler, standardized interface for applications.
* **Security and Protection**: System calls ensure that user applications cannot directly access hardware, which enhances system security and stability.
* **Context Switching**: When a system call is made, a context switch occurs from user mode to kernel mode, allowing the operating system to execute the requested operation safely.
* **API (Application Programming Interface)**: The system call interface is essentially an API provided by the operating system that includes a set of functions available to user applications.

1. List the types of System Calls.

System calls can be categorized based on their functionalities. Here are the main types:

1. **Process Control**:
   * **Creation and Termination**: fork(), exec(), exit()
   * **Process Management**: wait(), kill()
   * **Attributes and Priority**: nice(), getpid(), setpriority()
2. **File Management**:
   * **File Operations**: open(), close(), read(), write()
   * **Directory Operations**: mkdir(), rmdir(), chdir()
   * **Metadata Operations**: stat(), chmod(), chown()
3. **Device Management**:
   * **Device Operations**: ioctl(), read(), write()
   * **Buffer Management**: request\_buffer(), release\_buffer()
4. **Information Maintenance**:
   * **System Information**: uname(), gettimeofday()
   * **User Information**: getuid(), getgid()
   * **Accounting**: acct()
5. **Communication**:
   * **Interprocess Communication (IPC)**: pipe(), shmget(), shmat()
   * **Message Passing**: msgget(), msgsnd(), msgrcv()
   * **Network Communication**: socket(), connect(), send(), recv()

These categories encompass a wide range of functionalities that enable applications to effectively utilize system resources and perform necessary operations.

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