EXPERIMENT-1

Create a new process by invoking the appropriate system call. Get the process identifier of the currently running process and its respective parent using system calls and display the same using a C program.

AIM:-

To create a new process using system calls, retrieve the process identifier (PID) of the currently running process and its parent process (PPID), and display them.

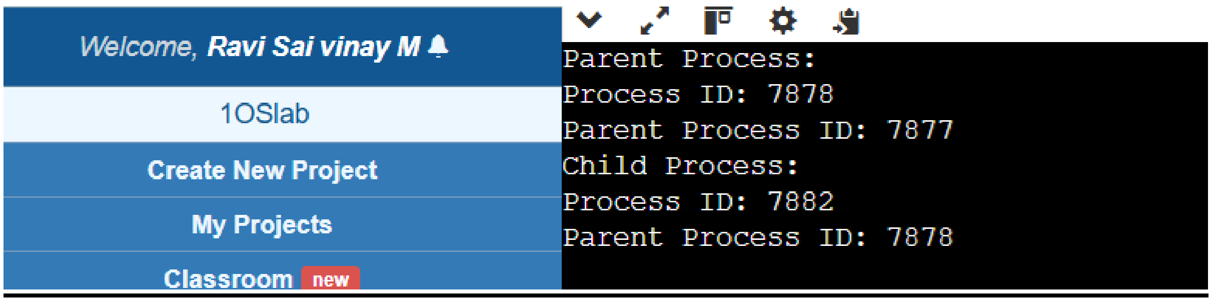
ALGORITHM:-

1. Start the program.
2. Use the fork() system call to create a new process.
    - If fork() returns 0, it indicates the child process.
    - If fork() returns a positive value, it indicates the parent process.
3. Use the getpid() system call to get the PID of the process.
4. Use the getppid() system call to get the PID of the parent process.
5. Print the PID and PPID for both the parent and child processes.
6. End the program.

CODE:-

```
#include <stdio.h>

#include <unistd.h>

#include <sys/types.h>

int main() {

    pid_t pid;

    // Create a new process

    pid = fork();
    if (pid < 0) {
        // Fork failed
        perror("Fork failed");
        return 1;
    } else if (pid == 0) {
        // Child process
        printf("Child Process:\n");
        printf("Process ID (PID): %d\n", getpid());
        printf("Parent Process ID (PPID): %d\n", getppid());
    } else {
        // Parent process
        printf("Parent Process:\n");
        printf("Process ID (PID): %d\n", getpid());
        printf("Parent Process ID (PPID): %d\n", getppid());
    }
    return 0;
}
```

OUTPUT:-



```
Parent Process:
Process ID: 7878
Parent Process ID: 7877
Child Process:
Process ID: 7882
Parent Process ID: 7878
```

RESULT:-

A new process was successfully created using the fork() system call. The process identifiers (PIDs) of both the parent and child processes, as well as the parent's process identifier (PPID), were displayed.