#### System Calls 1

**Aim**

Write programs for illustrating the working of the following system calls in Linux : **fork(), getpid(), getppid(), execv(), wait(), exit()**

**Description:**

**fork():** This system call is used to create a new process, which is called the child process, which is an exact copy of the calling process, called the parent process. After a fork, both the parent and the child processes continue to execute the same program, but they have different process IDs.

**getpid()**: This system call is used to get the process ID (PID) of the current process. Every process in a UNIX-like operating system is identified by a unique PID.

**getppid()**: This system call is used to get the process ID of the parent of the current process, known as the parent process ID (PPID). Each process, except for the initial process (usually PID 1), has a parent process.

**execv()**: This system call is used to execute a program. It replaces the current process image with a new process image by loading a new program into the current process's memory space. It takes the path to the executable file and an array of pointers to null-terminated strings that represent the program's arguments.

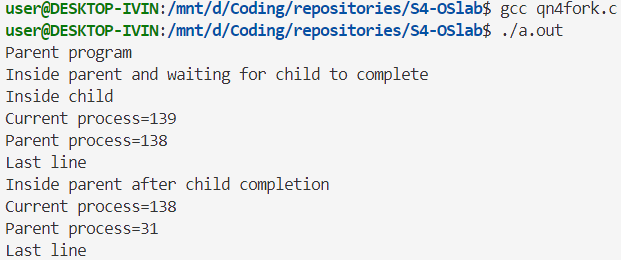
**wait()**: This system call is used by a parent process to wait for the termination of its child processes. It suspends the execution of the parent process until one of its child processes exits. It also allows the parent process to retrieve the exit status of the terminated child process.

**exit()**: This system call is used to terminate the current process. It takes an exit status as an argument, which is typically used to indicate the success or failure of the process. The exit status is available to the parent process via the wait() system call.

**Program 1**

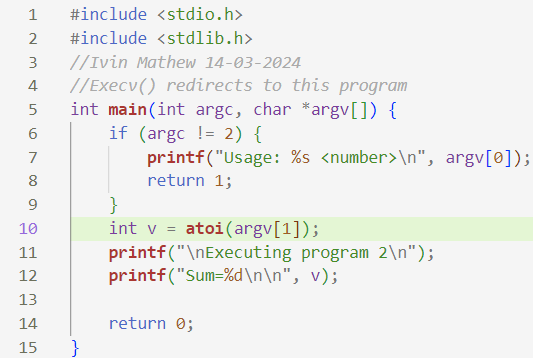
****

**Sample input and output**

****

**Program 2**

****

****

**Sample input and output**

****

**Result**

The programs have been executed and output has been successfully verified.