|  |  |  |  |
| --- | --- | --- | --- |
| **Name of Student** |  | | |
| **Lab Experiment No.** | 3.1 | **Roll No.** |  |
| **Date Of Perf.:** |  | **Date Of Sub.:** |  |
| **Expt. Title** | To study process management in OS (System calls and Unix commands) | | |
| **CO Mapping** | LO1,LO2,LO5 | | |

**Aim:** To study process management in OS using system calls.

**Objectives of the Experiment:**

To study system calls fork, getpid, getppid, wait, sleep...

**Theory:**

* 1. **System Call : int fork()**

System call **fork()** is used to create processes. It takes no arguments and returns a process ID. The purpose of **fork()** is to create a ***new*** process, which becomes the *child* process of the caller. After a new child process is created, ***both*** processes will execute the next instruction following the ***fork()*** system call.

* If **fork()** returns a negative value, the creation of a child process was unsuccessful.
* **fork()** returns a zero to the newly created child process.
* **fork()** returns a positive value, the ***process ID*** of the child process, to the parent. The returned process ID is of type **pid\_t** defined in **sys/types.h**. Normally, the process ID is an integer. Moreover, a process can use function **getpid()** to retrieve the process ID assigned to this process.

1. **System Call: int getpid(), int getppid()**

getpid() and getppid() return a process's id and parent process's id numbers, respectively.

1. **System Call: int exit(int status)**

exit() closes all of a process's file descriptors, deallocates its code, data, and stack, and then terminates the process.

1. **System Call: int wait(int\* status)**

wait() causes a process to suspend until one of its children terminates. A successful call to wait() returns the pid of the child that terminated and places a status code into status

**Terminology:**

* 1. **Orphan Processes**

If a parent dies before its child, the child is automatically adopted by the original "init" process, PID 1.

* 1. **Zombie processes**

When a child process terminates, it sends its parent a SIGCHLD signal and waits for its termination code status to be accepted. A process that is waiting for its parent to accept its

return code is called a zombie process.

|  |
| --- |
| 1. Print “Hello world” 2 time using fork system call. 2. Print “hello world” 9 times using fork system call 3. Print process Id and Parent process id of child process. |
| **Post Lab Assignment:** |
| Explore following commands with examples   * fg ,bg, stop, jobs, at * batch, nohup,nice, kill |