

**Faculty of** **Technology and Engineering**

# U & P U. Patel Department of Computer Engineering

Date: 14 / 08 / 2022

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Academic Year | : | 2022-23 | Semester | : | 5 |
| Course code | : | CE354 | Course name | : | Operating System |

PRACTICAL – 7

**AIM: Aim: The demonstration and use of execve() and wait() system call along with zombie and orphan state.**

|  |  |
| --- | --- |
| Zombie Process: | A process which has finished the execution but still has entry in the process table to report to its  parent process is known as a zombie process. A child process always first becomes a zombie before  being removed from the process table. The parent process reads the exit status of the child process  which reaps off the child process entry from the process table.  In the following code, the child finishes its execution using exit() system call while the parent sleeps  for 50 seconds, hence doesn’t call wait() and the child process’s entry still exists in the process  table. |
| **Output:** |  |
| Orphan Process: | A process whose parent process no more exists i.e. either finished or terminated without waiting  for its child process to terminate is called an orphan process.  In the following code, parent finishes execution and exits while the child process is still executing  and is called an orphan process now.  However, the orphan process is soon adopted by init process, once its parent process dies. |
| **Output:** |  |
| **Output:** |  |
| Execvchild.c |  |
| **Output:** |  |
| **Question 1:** | Differentiate fork and execve system calls |
| **Answer:** | Fork starts a new process which is a copy of the one that calls it, while exec replaces the current process image with another (different) one. Both parent and child processes are executed simultaneously in case of fork() while Control never returns to the original program unless there is an exec() error. |