CSD 204 Lab

Lab 2: Process control system calls

[Dr. Sweta Kumari, Assistant Professor, SNIoE**]**

Deadline - 10th Feb 2025, 11:59PM

Goal: This lab will deepen your understanding of various concepts of process management in Linux, specifically on system calls such as FORK(), EXEC(), WAIT(), and EXIT().

You need to write a simple program using these system calls, as a warmup to solving this lab exercise. This is the demonstration of FORK(), EXEC(), WAIT(), and EXIT() system calls along with zombie and orphan states.

Before you begin

- Familiarize yourself with the various process related system calls in Linux: FORK(), EXEC(), WAIT(), and EXIT(). The "man pages" in Linux are a good source of learning. You can access the man pages from the Linux terminal by typing man fork, man 2 fork and so on.

- It is important to understand the different variants of these system calls. In particular, there are many different variants of the exec and wait system calls; you need to understand these to use the correct variant in your code. For example, you may need to invoke wait differently depending on whether you need to block for a child to terminate or not.

- Familiarize yourself with simple shell commands in Linux like echo, cat, sleep, ls, ps, top, grep and so on. To implement these commands in your shell, you must simply "exec" these existing executables, and not implement the functionality yourself.

Exercises

1. Write a C/C++ program to create child processes of a main process using fork system call for N times. N can be any number ranging from 1 to 10. Also print a message with each process saying "Hello I am a child process\ parent process. My Pid is ="__". Here, the child process should print "I am a child process" and exit. The parent should wait to reap the child, print a message "I am a parent process" after reaping the child, and then exit. Analyze the number of child processes while varying the value of N & mention in the report.

   (**Note** - Use getpid() to get the Pid of a process)

2. Write a C/C++ program in which the main program accepts the integers to be sorted. Main program uses the FORK system call to create a new process called a child process. Parent process sorts the integers using a sorting algorithm and waits for the child process using WAIT system call to sort the integers using any sorting algorithm. Also demonstrate zombie and orphan states.
3. Write a C/C++ program in which the main program accepts an integer array. Main program uses the FORK system call to create a new process called a child process. Parent process sorts an integer array and passes the sorted array to the child process through the command line arguments of EXEC system call. The child process uses EXEC system call to load a new program that uses this sorted array for performing the binary search to search for the particular item in the array.

Submission Format:- You have to upload: (1) The source code in the following format: Assgn2Src-<Name>.c, also include the instructions for executing the program in the comment. (2) Report: Assgn2Report-<Name>.pdf. Name the zipped document as: Assgn2-<Name>.zip.

Note: Please follow this naming convention mentioned above. make different files for codes for each question.

Grading Policy: - The policy for grading this assignment will be - (1) C/C++ code of the above problems: 75% (2) Analysis of the results: 20% (3) Code documentation and indentation: 5%.

Please note:
- All assignments for this course have a late submission policy of a penalty of 10% each day after the deadline of six days. After that, it will not be evaluated.
- **All submissions are subject to plagiarism checks. Any case of plagiarism will be dealt with severely.**