Operating Systems - Assignment 1

September 27, 2020

Exercise 1.1: Process creation and termination system calls (Total points: 50).

Due date: Sept 30, 2020. Time: 23:59 Hrs. (Hard Deadline)

The first exercise deals using process creation system call, `fork()`. You need to write a program that spawns a child process, using the `fork()` system call. The child process reads a CSV file, presented with the assignment, that has (fake) student IDs and grades of various assignments. This child process computes the average score of each student for section 'A', and thereafter prints the details of these students (of section 'A', *i.e.*). The parent process does the same operation, on the same CSV file but for students of section 'B'.

The parent process must wait for the child process to terminate, by using the system call `waitpid()`. The child process must call the `exit()` system call once its execution ends.

You would require to refer to the `manpages` for various system calls mentioned.

Note: You are expected to use `read` and `read` system calls, to read and write to the file.

What To Submit

- Program source code with Makefile to compile and pause the compilation at each phase.
- Write-up describing the how your program works, with details of each system call, the arguments passed and the expected outcomes and how you handled error/corner cases.

Grading Rubric

- Successful compilation of the program via the Makefile – 10 points.
- Correct output for the program, corresponding to each of the sections – 20 points.
- Correct use the system calls mentioned with proper arguments and error handling – 15 points.
- Descriptive write-up – 5 points.

Exercise 1.2: Basic Linux/Unix shell (Total points: 50).

Due date: Sept 30, 2020. Time: 23:59 Hrs. (Hard Deadline)

Linux (and other Unix like OSes), have "shells" or programs which present a command line interface to users to type commands in. In this assignment you need to use standard C libraries, including Linux system calls such as `fork()`,`exec()` family system calls and `wait()` family of system calls.

There are two kinds of commands – "internal" and "external". Internal commands are those which are interpreted by the shell program itself, without requiring a different program to handle the expected operations (of the said command). Examples of internal commands are like `'cd'`, `'pwd'`, `'exit'` *etc.* External commands on the other hand relate to commands which are not handled directly by the shell program but by an external program. Common examples include `'ls'`, `'cat'`, `'grep'` *etc.*

Your task is to design your a simple shell that can handle **five**, internal commands – `'cd', 'echo', 'history', 'pwd'` and `'exit'`. These commands would be handled directly by the shell. Your shell should also be able to handle **five** external commands – `'ls', 'cat', 'date', 'rm' and 'mkdir'`. For these external commands you need to write individual programs to handle these commands. To handle these external commands, the shell should typically create a new process, using the `fork()` system call and within each process you need to use the `execl()` family system call to run the individual program. The parent program must also wait for the child program to terminate using the `wait()` family of system calls.

For each of these commands, you need not handle all the command line options. Two options per command is sufficient. You need to document which two options you are handling and need to demonstrate correct functioning of the command with respect to (atleast) your chosen options. You also need to handle corner cases such as invalid options (graceful degradation).

What To Submit

- The C program sources.
- `Makefile` to compile the source and generate the running binary for the shell.
- Write-up describing the system, the options the shell commands can take, the errors you handled and the assumptions you made. Also provide test cases to test the functioning of the shell.

Grading Rubric

- Successful compilation using Makefile – 5 points.
- Use of system calls like – `fork()`, `execl()` family of system calls, `wait()` family of system calls to handle external commands – 10 points.

- Correct handling of commands (two options per command, as described earlier) – 20 points.
- Successfully handling atleast two corner cases for each of the commands – 20 points (List the bugs/errors/attacks that you defend against).
- Description of the systems, commands to execute and test the program and the assumptions that you made – 5 points.