New York University Tandon School of Engineering - Introduction to Operating Systems - Assignment 4

Fall 2024

Assignment 4 (10 points)

This assignment contains 10 problems.

Problem 1 (2 points)

If you create a `main()` routine that calls `fork()` twice, then `execlp()` to run the `ls` command, and then calls `fork()` again, i.e., if it includes the following code:

```
pid_t x=-11, y=-22;
x = fork();
if(x==0) {
    execlp("ls","ls",NULL);
    perror("execlp failed");
    exit(1);
}
if(x>0) y = fork();
```

Assuming all `fork()` calls succeed, and `execlp` does not fail, draw a process tree similar to that of Fig. 3.8 (page 116) in your textbook, clearly indicating the values of x and y for each process in the tree (i.e., whether 0, -11, -22, or larger than 0). The process tree should only have one node for each process, and each line/arrow in the process tree diagram should represent a creation of a process, or alternatively a parent/child relationship. The process tree should be a snapshot just after all forks completed but before any process exits. *(Insert process tree diagram here)*

Problem 2 (4 points)

Write a program that creates the process tree shown below. The program should use only `fork()`. The process tree should show the PID of each process. *(Insert process tree diagram here - A tree with a parent and three children)*

Problem 3 (4 points)

Write a program whose `main()` routine obtains a parameter `n` from the user (i.e., passed to your program when it was invoked from the shell, $n > 0$) and creates a child process. The child process shall then calculate the factorial of `n` and print the result. The parent waits for the child to exit and then prints the square of the factorial calculated by the child. Do not use IPC in your solution to this problem (i.e., neither shared memory nor message passing).

Problem 4 (2 points)

Explain the difference between a zombie process and an orphan process. Give an example code snippet that could create each.

Problem 5 (3 points)

Describe a scenario where using `exec()` family of functions would be preferable to forking and running a program directly within the child process.

Problem 6 (2 points)

What is the purpose of the `wait()` system call? What happens if a parent process doesn't call `wait()` for its child process?

Problem 7 (3 points)

Write a C program that uses `fork()` to create two child processes. Each child process should print its own process ID and the process ID of its parent. The parent process should wait for both child processes to finish before exiting.

Problem 8 (2 points)

Illustrate with a code example how to use signals to gracefully handle interruptions (e.g., Ctrl+C) in a long-running process.

Problem 9 (3 points)

Write a program that uses pipes to pass a string from a parent process to a child process. The child process should receive the string, reverse it, and print the reversed string to the standard output.

Problem 10 (3 points)

Explain the concept of process scheduling and briefly describe two common scheduling algorithms (e.g., Round Robin, Shortest Job First). Discuss their strengths and weaknesses.

What to hand in (using Brightspace)

Please submit the following files individually:

1. Source file(s) with appropriate comments. The naming should be similar to "`lab#_$_c`" (# is replaced with the assignment number and $ with the question number within the assignment, e.g., `lab4_b.c`, for lab 4, question b OR `lab5_1a` for lab 5, question 1a).
2. A single pdf file (for images + report/answers to short-answer questions), named "`lab#.pdf`" (# is replaced by the assignment number), containing:
    - Screenshot(s) of your terminal window showing the current directory, the command used to compile your program, the command used to run your program, and the output of your program.
3. Your Makefile, if any. This is applicable only to kernel modules.

RULES

- You shall use kernel version 4.x.x or above. You shall not use kernel version 3.x.x.
- You may consult with other students about GENERAL concepts or methods but copying code (or code fragments) or algorithms is NOT ALLOWED and is considered cheating (whether copied from other students, the internet, or any other source).
- If you are having trouble, please ask your teaching assistant for help.
- You must submit your assignment prior to the deadline.