New York University
Tandon School of Engineering
Department of Computer Science and Engineering
Introduction to Operating Systems
Fall 2024
Assignment 4 (10 points)

1) A) (2 points)

If you create a main() routine that calls `fork()` twice, followed by a `fork()` in the child process, i.e. if it includes the following code:

```
pid_t x=-11, y=-22;
x = fork();
if(x==0) y = fork();
if(y==0) fork();
```

Assuming all `fork()` calls succeed, 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). Note that the process tree should only have one node for each process and thus the number of nodes should be equal to the number of processes. The process tree should be a snapshot just after all forks completed but before any process exists. Each line/arrow in the process tree diagram shall represent a creation of a process, or alternatively a parent/child relationship.

1) B) (4 points)

Write a program that creates the process tree shown below: *(A process tree diagram showing a parent process forking three times, resulting in a total of 4 processes in the tree would be inserted here.)*

2) (4 points)

Write a program whose main routine obtains two parameters, a and b, from the user (i.e., passed to your program when it was invoked from the shell, a and b are positive integers). The program creates a child process. The child process calculates a to the power of b (a¡sup¿b¡/sup¿) and prints the result. The parent process waits for the child to finish and then prints the sum of a and b. Do not use IPC in your solution. Handle potential errors such as invalid input (e.g., non-positive integers).

3) (2 points)

Explain the difference between a zombie process and an orphan process. Give an example scenario that could lead to the creation of each.

4) (2 points)

Describe a situation where using `exec()` system calls would be preferable to forking a new process and running the code within the child process. Explain your reasoning.

5) A) (2 points)

If a process calls `fork()` and then immediately calls `exit()`, what happens to the child process, if any? What if the parent calls `exit()` first?

5) B) (2 points)

What is the purpose of the `wait()` system call? Explain how it works and what problems it helps to avoid.

6) (4 points)

Write a C program that creates two child processes. The first child process calculates the factorial of a number n (provided as a command-line argument). The second child process calculates the sum of numbers from 1 to n. Both children should print their results. The parent process waits for both children to finish and then prints the difference between the factorial and the sum. Error handling for invalid input (e.g., non-positive n) is required. Do not use IPC.

7) (4 points)

Write a C program that simulates a simple producer-consumer problem using only fork() and wait(). The producer generates 5 random numbers (between 1 and 100) and stores them in a shared array (global variable). The consumer reads these numbers from the array and prints their sum. Ensure proper synchronization using wait() and its variants to avoid race conditions. (Hint: The producer might use a counter to signal how many numbers are produced.)

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 c 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 form 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.