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

Problem 1 (2 points)

If you create a main() routine that calls fork() four times, i.e., if it includes the following code:

```
pid_t w=-1, x=-2, y=-3, z=-4;
w = fork();
if(w==0) x = fork();
if(x==0) y = fork();
if(y>0) z = 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 w, x, y, and z for each process in the tree (i.e., whether 0, -1, -2, -3, -4, 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.

(Include your process tree diagram here as an image using *problem1_tree.png* *)*

Problem 2 (4 points)

Write a program that creates the process tree shown below:

(Include the process tree diagram here as an image using problem2_tree.png*)*

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¿2) and creates two child processes. The first child process shall create and print a Fibonacci sequence of length n/2 (integer division) whose elements are of type `unsigned long long`. The second child process shall print the next n/2 elements of the Fibonacci sequence. You may find more information about Fibonacci numbers at (https://en.wikipedia.org/wiki/Fibonacci_number). The parent waits for both children to exit and then prints a message indicating completion. Do not use IPC in your solution to this problem (i.e., neither shared memory nor message passing).

(Include your C code here)

Listing 1: problem3.c

```
// Your C code for problem 3 goes here
```

Problem 4 (2 points)

Explain the difference between a process and a thread. Give examples of when you might prefer to use threads over processes, and vice versa.

Problem 5 (3 points)

Describe the concept of a zombie process. How are zombie processes created, and what are the potential problems they can cause? How can they be avoided?

Problem 6 (1 point)

What is the purpose of the ‘wait()‘ system call?

Problem 7 (2 points)

Draw a process state diagram showing the possible transitions between states (e.g., Running, Ready, Blocked).

Problem 8 (2 points)

Explain the concept of context switching and its importance in operating systems.

Problem 9 (2 points)

What are the benefits and drawbacks of using threads?

Problem 10 (2 points)

Briefly describe the differences between the following scheduling algorithms: First-Come, First-Served (FCFS), Shortest Job First (SJF), and Round Robin. Which algorithm is generally considered to be the fairest, and why?

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.