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

Problem 1 (10 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.

Problem 2 (10 points) Write a program that creates the following process tree:

```
     A
    / \
   B   C
  / \   \
 D   E   F
```

Where A is the initial process. Clearly label each process in your code.

Problem 3 (10 points) Write a program whose main routine obtains parameters n and m from the user (passed to your program when invoked from the shell, n¿2, m¿0). The main process creates m child processes. Each child process calculates the factorial of n using `unsigned long long` and prints it along with its process ID. The parent process waits for all children to finish and then prints the sum of all factorials calculated by the children. Do not use IPC.

Problem 4 (10 points) Consider a scenario where a process needs to create a tree of processes with a depth of 3. Each node in the tree represents a process. The root process (level 0) creates two child processes (level 1). Each level 1 process creates two child processes (level 2), and each level 2 process creates one child process (level 3). Design a C program to achieve this using ‘fork()‘. Each process should print its level and process ID. The parent process should wait for all its descendants to finish before exiting.

Problem 5 (10 points) Write a program that uses fork() to create three processes. The parent process sends a signal to each child process. Each child process prints the signal it received and its process ID. The parent process then waits for all child processes to terminate.

Problem 6 (10 points) Write a C program that uses 'fork()' to create a process tree where the root process forks two children. Each of those children then forks two more children each. Each process should print its level in the tree (0 for the root, 1 for its children, 2 for its grandchildren) and its process ID before exiting. The parent process waits for all its descendants to terminate before exiting itself.

Problem 7 (10 points) Design a program that takes an integer N as command-line input. The main process creates N child processes. Each child process calculates the sum of the squares of numbers from 1 to its process ID (using unsigned long long). The parent process waits for all child processes to complete, then calculates the average of the sums calculated by its children, printing the result to the console. Do not use any form of inter-process communication.

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.

Operating Systems - Prof. Omar Mansour