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() twice, i.e., if it includes the following code:

```
pid_t x=-11, y=-22;
x = fork();
if(x==0) y = 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). The process tree should only have one node for each process. The process tree should be a snapshot just after all forks completed but before any process exits. Each line/arrow in the process tree diagram shall represent a creation of a process, or alternatively a parent/child relationship.

Note: A graphical process tree should be included here. I cannot create images in LaTeX.

Problem 2 (4 points)

Write a program that creates the process tree shown below. The process tree should have a parent process that forks three child processes. Each of these three children then forks two children each.

Note: A graphical process tree diagram would be included here, but I cannot create images in LaTeX.

Problem 3 (4 points)

Write a program whose main routine obtains two parameters, n and m, from the user (passed to your program when invoked from the shell, n¿1, m¿1) and creates two child processes. The first child process shall calculate the factorial of n (n!), and the second child process shall calculate the factorial of m (m!). Both factorials should be of type `unsigned long long`. The parent waits for both children to exit and then prints the sum of n! and m!. 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 of how each might arise.

Problem 5 (3 points)

Describe three different ways to terminate a process in a Unix-like operating system. Provide example commands for each method.

Problem 6 (2 points)

What is the purpose of the wait() system call? Explain what happens if a parent process does not call wait() for its child processes.

Problem 7 (2 points)

Draw a process state diagram showing the possible states a process can be in and the transitions between those states. Include at least five states.

Problem 8 (3 points)

Write a C program that uses the exec family of functions to replace the current process image with the ‘ls -l‘ command. Show the code and explain how the exec function works.

Problem 9 (3 points)

Describe the concept of a process control block (PCB). What information does it typically contain? Why is it crucial for operating system functionality?

Problem 10 (3 points)

Write a program that creates five child processes. Each child process should sleep for a random number of seconds between 1 and 5, then print its process ID and the message "Child process finished." The parent process should wait for all child processes to finish before printing "All children finished."

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.