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

September 12, 2025

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). 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.

(Insert process tree diagram here - Problem 1)

Problem 2 (4 points)

Write a program that creates the process tree shown below:

(Insert process tree diagram here - Problem 2)

Problem 3 (4 points)

Write a program whose main routine obtains a parameter n from the user (passed to your program when invoked from the shell, n¿2) and creates a child process. The child process shall then calculate and print the sum of all even numbers from 1 to n. The parent waits for the child to exit and then prints the sum of all odd numbers from 1 to n. 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. Provide examples of how each might arise.

Problem 5 (3 points)

Consider a scenario where a parent process creates three child processes. Each child process performs a different task:

- Child 1: Calculates the factorial of a number provided as a command-line argument.
- Child 2: Finds the largest prime number less than a number provided as a command-line argument.
- Child 3: Sorts an array of numbers provided as a command-line argument.

Describe how you would design the parent process to manage these children, including error handling and waiting for completion.

Problem 6 (2 points)

Draw a process tree that results from the following code:
c pid$_t pid1, pid2, pid3$;
pid1 = fork(); if (pid1 == 0) pid2 = fork(); if (pid2 == 0) pid3 = fork();

Problem 7 (2 points)

What is the purpose of the ‘wait()‘ system call? What happens if a parent process does not call ‘wait()‘ after creating a child process?

Problem 8 (2 points)

Explain the concept of process states (e.g., running, ready, blocked). Draw a state diagram illustrating the transitions between these states.

Problem 9 (3 points)

Write a C program that creates a child process. The parent process sends a message to the child process using pipes. The child process receives the message, reverses it, and sends the reversed message back to the parent. The parent process then prints the reversed message.

Problem 10 (2 points)

Describe the differences between process and thread. When would you prefer to use threads over processes, and vice versa?

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.