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

Operating Systems - Prof. Omar Mansour

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=-11, x=-22, y=-33, z=-44;
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, -11, -22, -33, -44, 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 exits. 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 2 (4 points)

Write a program that creates the process tree shown below:

[Insert Process Tree Diagram Here]

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 a child process. The child process shall then calculate the sum of the first n even numbers and print the result. The parent waits for the child to exit and then prints the sum of the first n+2 even numbers. 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 process and a thread. Provide examples of when you would 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 you avoid creating zombie processes?

Problem 6 (2 points)

Draw a process state diagram illustrating the transitions between the different states a process can be in (e.g., new, ready, running, blocked, terminated). Label each transition with the event that causes it.

Problem 7 (3 points)

Write a C program that uses the 'execl' system call to execute the 'ls -l' command. The program should then print a message indicating that the 'ls -l' command has finished executing. Handle potential errors appropriately.

Problem 8 (2 points)

What is a race condition? Give a concrete example of a race condition in a multi-threaded program, and explain how to prevent it using appropriate synchronization mechanisms (e.g., mutexes, semaphores).

Problem 9 (4 points)

Write a program that uses pipes to implement a simple inter-process communication mechanism. The parent process should send a string to the child process through a pipe, and the child process should receive the string, modify it (e.g., append a suffix), and send it back to the parent process through another pipe. The parent process should then print the modified string.

Problem 10 (3 points)

Explain the differences between shared memory and message passing as inter-process communication mechanisms. Discuss the advantages and disadvantages of each approach, considering factors like performance, complexity, and data synchronization.

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.