NYU Tandon School of Engineering - Introduction to Operating Systems - Assignment 4

Fall 2024

Assignment 4 (7 problems, 40 points)

Problem 1 (3 points)

If you create a main() routine that calls fork() twice, as follows:

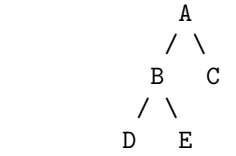
```
pid_t x=-11, y=-22;
x = fork();
if(x==0) y = fork();
```

Assuming all fork() calls succeed, draw a process tree, clearly indicating the values of x and y for each process.

[Insert Process Tree Diagram Here]

Problem 2 (5 points)

Write a C program that creates the following process tree:



Each process should print its own process ID and its parent's process ID.

Problem 3 (4 points)

Write a C program that forks a child process. The parent process sends a single integer (e.g., 10) to the child process using a pipe. The child receives the integer, adds 5 to it, and then prints the result. The parent waits for the child to finish.

Problem 4 (6 points)

Write a program that creates a process tree with three levels. The root process (level 0) creates two child processes (level 1). Each level 1 process then creates two child processes (level 2). Each process should print its level and process ID. Use only fork().

Problem 5 (6 points)

Modify the program from Problem 3 to use shared memory instead of a pipe for communication between the parent and child. The parent should write the integer to shared memory, the child should read it, add 5, and print the result. The parent should then read the result from shared memory and print it as well. Ensure proper synchronization using semaphores to prevent race conditions.

Problem 6 (8 points)

Write a program that simulates a simple producer-consumer problem using two processes and shared memory. The producer generates random numbers (between 1 and 100) and writes them to the shared memory buffer. The consumer reads numbers from the buffer and prints them. The buffer should have a limited size (e.g., 5). Use semaphores for synchronization. Handle full and empty buffer conditions appropriately.

Problem 7 (8 points)

Write a program whose main routine obtains parameters n and m from the user (passed as command-line arguments, n ¿ 2, m ¿ 0) and creates m child processes. Each child process calculates and prints the nth Fibonacci number, using `unsigned long long` for the calculation. The parent process waits for all children to complete and then prints the sum of all calculated Fibonacci numbers. Do not use IPC in this problem.

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.