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

Your Name

September 12, 2025

1 Problem 1: Process Tree (2 points)

1. Draw a process tree here. You'll need to manually create a tree diagram showing the parent-child relationships. The tree should represent the following scenario: Process A forks into B and C. Process B forks into D and E. Process C forks into F. Each node in the tree should be labeled with the process name (A, B, C, D, E, F). Each arrow/line should indicate a parent-child relationship.

2 Problem 2: Process Tree Program (4 points)

1. Write your C code here to generate the process tree specified in Problem 1. Remember to include appropriate comments. Save this as ‘lab4$_b$*.c‘.Theprogramshouldcreatetheprocesses*
*childrelationshipsasdescribedinProblem*1*.EachprocessshouldprintitsprocessIDtotheconsole.*

3 Problem 3: Producer-Consumer (4 points)

1. Write a C program that uses two processes, a producer and a consumer, to simulate a bounded buffer. The producer generates random integers between 1 and 100 and adds them to the buffer. The consumer removes inte-

gers from the buffer and prints them. The buffer size is 5. Use semaphores for synchronization. Save this as 'lab4$_c$.c'.

4 Problem 4: Shared Memory (4 points)

1. Write a C program that uses shared memory to allow two processes to share a counter. One process increments the counter 1000 times, and the other process decrements the counter 1000 times. Use appropriate synchronization mechanisms to prevent race conditions. Print the final value of the counter. Save this as 'lab4$_d$.c'

5 Problem 5: File Locking (4 points)

1. Write a C program that uses file locking to ensure that only one process can write to a file at a time. Two processes should attempt to write to the same file concurrently. One process writes "Hello", the other writes "World". The output should be either "Hello" or "World", but not a corrupted mix of both. Save this as 'lab4$_e$.c'.

6 Problem 6: Zombie Processes (2 points)

1. Explain what a zombie process is and how it can be avoided in C. Provide a code snippet demonstrating a situation where a zombie process might be created and a corrected version that avoids this.

7 Problem 7: Signal Handling (4 points)

1. Write a C program that handles the SIGINT signal (Ctrl+C). When SIGINT is received, the program should print a message "Caught SIGINT!" and then gracefully exit. Demonstrate that the signal handler is correctly invoked when Ctrl+C is pressed. Save this as 'lab4$_f$.c'.

8 What to Hand In

1. Source code files ('lab4$_b$.c', '$lab4_c.c$', '$lab4_d.c$', '$lab4_e.c$', '$lab4_f.c$').

1. A PDF file ('lab4.pdf') containing:
   - Screenshots of your terminal showing compilation and execution for each problem.
   - The process tree diagram from Problem 1.
2. Makefile (if applicable).

9 Rules

- Use kernel version 4.x.x or above.
- No copying code or algorithms.
- Ask your TA for help if needed.
- Submit on time.