COMP304 - Operating Systems: Assignment 1

Start: 7 Oct 2022

Due: 18 Oct 2022, Midnight

Notes: This is an individual assignment, partners are not allowed and cheating is a punishable offense. You may discuss the problems with your peers but the submitted work must be your own work. No late assignment will be accepted. Submit your answers to Blackboard. This assignment is worth 4% of your total grade.

Corresponding TA: Mohammad Issa (missa18@ku.edu.tr) Office: ENG 230

Office Hours: 5:30-6:30 on Tuesday and Wednesday

Possible Points: 50 points

# Problem 1: Forks (15 pts: 7 + 8)

- Part 1:
  - The parent process creates a child process and the child process will output the time of the day and its own PID every second.
  - The parent process waits for 5 seconds.
  - The parent process will kill the child process after 5 seconds and prints "Child <PID> killed.". Then the parent terminates.
- Part 2:
  - The parent process creates 4 child processes and each child process will output the time of the day and its own PID every second.
  - The parent process waits for 5 seconds.
  - The parent process will kill the child processes after 5 seconds and prints "Child <PID> killed.". Then the parent terminates.

You may need to use sleep(), gettimeofday(), kill() and fork() system calls in your assignment. You are required to submit your C source code file along with a snapshot of the execution in terminal for each part.

# Problem 2: Pipes (15 pts: 9 + 6)

- Part 1
  - The parent process creates 2 children.
  - The parent process reads one number (integer) from the standard input. The parent process sends this number to the children using standard pipes.
  - The first child multiplies the number by its own PID, prints it as "First Child: Input <X>, Output <PID> * <X>" and sends the output back to the parent.
  - The second child divides its own PID by the input, prints it as "Second Child: Input <X>, Output <PID> / <X>" and sends the output back to the parent.
  - The parent sums both outputs and prints the final result.
  - The parent kills all children processes, and terminates
- Part 2
  - Add a sleep(5) parameter to each process operaion, so that you can track the behavior as it executes.
  - Open two terminals. In the first terminal, run the tool *htop*. Once in htop, press *t* to sort all processes as a process tree. Resize the window so that you can see the processes properly. In the second terminal, execute your program. Find the program in htop, and take a screenshot of its hierarchy (parent/child/grandchild).
  - Kill the first child process amidst execution by pressing *k* in htop, and sending it a SIGKILL signal (9) to immediately terminate it. Take a screenshot of the program execution output.

For Problem 2, you may need to use sleep(), kill() and fork() system calls. Refer to the ordinary pipe example in the textbook (also in Blackboard). You are required to submit your C source code file as well as a snapshot of the execution in terminal for each part.

# Problem 3: Shared Memory (20 pts)

Chinese whispers (kulaktan kulağa) is a popular children's game in which the first player comes up with a message, whispers in the ear of the second player, who then whispers the message into the ear of the third player and so on. This carries on until the last player receives the message who then reads the message aloud. Most often than not, the message that is read is different than the original message by the first player.

In this problem, you will write a code that will simulate the Chinese whispers game using shared memory. You are required to implement two C programs; the first will be a shared-memory consumer-producer program while the second one will be a driver program. These two programs need to function as follows:

## The driver program

- This is the program that simulates the Chinese whisper by creating N processes.
- The program accepts three command line arguments; the executable name of the "consumer-producer program", an integer N (number of processes), and a string message
- The program then forks N many children, each of which executes the consumer-producer program with necessary arguments to play the game.
- The number of processes should be at least 2. Otherwise, the program should terminate.
- Note that the driver program must create a new child only after the current child terminates to prevent concurrent writes to the shared memory.

## The consumer-producer program

- If this program is run by the first child of the driver program, then it creates a shared memory segment and writes the original message into it.
- Otherwise, the program reads the message in the shared segment, randomly chooses two characters in the message, swaps those two characters and then writes back the distorted message in the shared segment. If it is run by the last child, then it also unlinks the segment.

Here is a sample output:

```
Parent: Playing Chinese whisper with 4 processes.
Child 1: The OS assignment deadline is approaching.
Child 2: The iS assignment deadline is approaching.
Child 3: The iS assignment aeadline is approdching.
Child 4: The iS hssignment aeadline Os approdching.
Parent terminating...
```

To get started for this problem, refer to the producer and consumer shared memory example codes on Blackboard for Lecture 4/5.

# Submission

You are required to submit the followings packed in a zip file (named your-username(s).zip) to Blackboard:

- Each part's files must be contained within different folders.
- .c source code files that implement each assignment. Please comment your implementation.
- Any supplementary files for your implementations if used (e.g. Makefile).
- Do not submit any executable files (a.out) or object files (.o) to Blackboard.
- Each student must create a github repository for the assignment and add the TA as a contributor (username is 'mktip'). We will be checking the commits during the course of the assignment and during the assignment evaluation. This is useful for you too in case if you have any issues with your OS.
- Insufficient amount of detail in the code repository (e.g. one commit only on the day of the deadline) will result in a 10 point penalty in the assignment grade.
- You should keep your github repo updated from the start to the end of the assignment. You should not commit the assignment at once when you are done with it; instead make consistent commits so we can track your progress.
- Selected submissions may be invited for a demo session. Thus each student is expected to be fully knowledgeable of the entire implementation.

Good Luck and Start Early