CEG 4350/5350 Project
Summer 2024

Project plan is due on 6/17, Monday.
Final report is due on 7/17, Wednesday.

Create a producer process and a consumer process sharing data. The producer generates total 100 integer data randomly, and the consumer process reads all those data. The data generated by the producer could be stored in a file by the producer; and the data consumed (i.e., read) by the consumer could be stored in another file by the consumer.

The file of produced data and the file of consumed data could be printed to verify that the two processes have cooperated correctly; that means, each data item is not lost and not consumed more than once.

Interprocess communication (IPC) between producer and consumer (to share data) can be implemented using any two of the following four methods:

1. Use a *Pipe* to transfer 100 data from the producer to the consumer. Note: a Pipe (anonymous pipe, named pipe, or an equivalent) should be created using a system call, instead of piping the (standard) output of one process to the (standard) input of another.
    - Refer to Section 3.7.4 in 10th edition (or Section 3.6.3 in 9th edition).

2. Use either the direct message passing or indirect message passing (using a mailbox, a message queue, or a similar one) to transfer 100 data from the producer to the consumer.
    - Refer to Section 3.7.2 in 10th edition for an example of message passing (or Section 3.5.2 in 9th edition).

3. Use the sockets to transfer 100 data from the producer to the consumer.
   Note: The producer and consumer processes can be executed on the same machine or on different machines.
    - A reference book chapter for socket programming is posted on the Pilot (in the *Syllabus and Project Description* module).

4. Use the shared memory and semaphores for the implementation of the logical ring-buffer (that can store up to 10 data items) and the synchronization.
    - For the use of shared memory, refer to Sections 3.7.1 in 10th edition (and Section 3.5.1 in 9th edition).
    - For the use of semaphores, refer to Sections 7.3 and 7.4 in 10th edition (or Section 5.9 in 9th edition).

Programming Languages and Operating Systems:

- For each implementation of IPC method, any programming language can be used on any machine running any OS. In other others, different programming languages and OSs can be used for different IPC methods.

- Instead of creating and using two processes — one for producer and one for consumer — you can create two threads in a process, except for the socket-based communication method.

References:

- Neil Matthew and Richard Stones, *Beginning Linux Programming*, 4th edition, Wrox, 2007.
- John Shapley Gray, *Interprocess Communication in Linux: The Nooks and Crannies*, Prentice Hall, 2003.
- K. A. Robbins and S. Robbins, *Unix System Programming: Communication, Concurrency, and Threads*, Prentice Hall, 2003.
- W. Richard Stevens and Stephen A. Rago, *Advanced Programming in the UNIX Environment*, 3rd edition, Addison Wesley, 2013.
- For socket programming in Java, refer to the Chapter 4 of *Distributed Systems*: *Concepts and Design*, 5th edition, which is posted on Pilot.

Reporting:

- **Upload your project plan to Pilot by 6/17,** describing the IPC methods selected, programming language and OS to be used (for each IPC method selected), system calls to be used (for each IPC method selected).
- **Upload the final report to Pilot by 7/17**, including the description of design, source codes, execution result, and discussion.

In your final project report, you should include the following:

1. A summary describing what IPC methods are implemented; what languages and OS were used for the implementation of each IPC method used.
2. For each IPC method implemented,
   - *Description of design*: description the system calls used and the behavior of each process.
   - *Source code* (print out).
   - *Result*: printing of random integer values in the order they were generated by the Producer and saved in a file, and printing of random integer values in the order they were read/received and saved in another file by the consumer.
3. Overall Discussion: Any technical things you learned (doing the project) that you want to share with other students in this class. Any technical things that can demonstrate how much effort you made on the project.

Grading:

Programming project 30% {project plan 5%, design and description of design 5%, implementation and documentation 7%, correctness of programs 7%, discussion 6%}

Bonus scores:

- **If you implement 3 IPC methods, you are eligible for additional maximum 5%, such that your maximum project score would be 35%.**
- **If you implement 4 IPC methods, you are eligible for additional maximum 10%, such that your maximum project score would be 40%.**