San José State University
Department of Computer Engineering

CMPE 142
Operating Systems
Section 1

Spring 2021
Instructor: Ron Mak

Assignment #4

Assigned: Friday, February 26
Due: Friday, March 5 at 11:30 AM
Team assignment, 100 points max

Interprocess communication
This assignment will give your team practice using a named pipe (FIFO) and shared memory to perform interprocess communication (IPC). You may write your programs in either C or C++.

Part 1: IPC using named pipes
Write two programs, **`PipeProducer`** and **`PipeConsumer`**. Run the producer program multiple times and the consumer program multiple times in separate terminal windows, all simultaneously.

The first producer process should create a named pipe (FIFO) and write data values 101, 102, 103, etc. to it. The second producer process should write data values 201, 202, 203, etc. to the pipe. Other producer processes should write similar data values. Each process should also write the data values to its standard output.

Each consumer process should read the data values from the pipe and write each value to its standard output.

The producer program should take two command-line parameters, an id and the number of data values to write to the pipe. The producer process with id 1 should first remove the named pipe (**`unlink`**) in case it's left over from a previous run and then create a new pipe (**`mkfifo`**). The other producer processes should simply write to the pipe that process 1 created. After each producer process has written all its data values, it should write a done message to its standard output and then terminate normally.

For example, you can start three producer processes with these commands, each in a separate terminal window:

```
./producer 1 50
./producer 2 60
./producer 3 70
```

The consumer program should take an id as its single command-line parameter. After all the producer processes have completed writing and there are no more data values in the pipe, each consumer process should write a done message to its standard output and then terminate normally.

For example, you can start two consumer processes with these commands, each in a separate terminal window:

```
./consumer 1
./consumer 2
```

Note that the writers will block opening the named pipe until a reader has opened it.

Part 2: IPC using shared memory

Write two programs, **`ShmemProducer`** and **`ShmemConsumer`**. You will run the producer program multiple times and the consumer program multiple times in separate terminal windows, all simultaneously. Like the programs in Part 1, the producer program should take an id and the number of data values from the command line, and the consumer program should take an id from the command line.

The producer process with id 1 should create a named shared memory segment in which a circular buffer of a given size, say 5, will reside. Each producer process should write data values as in Part 1 into the buffer.

Each consumer process should read from the buffer in shared memory.

You will need named semaphores to synchronize the actions of the producers and consumers. Producer process 1 should also create these. Be sure to remove the semaphores left over from previous runs. You might also need one or more mutexes. One way to share a mutex among multiple processes is to recall that a mutex is a binary semaphore.

What to submit

Submit the following to Canvas, **Assignment #4: Interprocess Communication**.

- Source files (either C or C++) of your programs **`PipeProducer`**, **`PipeConsumer`**, **`ShmemProducer`**, and **`ShmemConsumer`**.
- For Part 1, a screenshot of three producers running (in three separate terminal windows) and two consumers running (in two separate terminal windows). Make a similar screenshot for Part 2.

Rubric

Your submission will be graded according to these criteria:

Criteria	Max points
Part 1	**40**
• Source file(s) of program **`PipeProducer`**	• 15
• Source file(s) of program **`PipeConsumer`**	• 15
• Screen shot of terminal windows running the producers and consumers.	• 10
Part 2	**60**
• Source file(s) of program **`ShmemProducer`**	• 25
• Source file(s) of program **`ShmemConsumer`**	• 25
• Screen shot of terminal windows running the producers and consumers.	• 10