**CS 354 Fall 2025**

**Lab 3: Coordination of Processes [20 pts]**

**Due: 09/22/2025 (Mon), 11:59 PM**

**Source code**

Start from a fresh copy of XINU xinu-fall2025.tar.gz by running the command:

tar zxvf /homes/cs354/xinu-fall2025.tar.gz

**The Problem To Be Solved**

Consider two processes that need to pass information between them. To make the

situation easy, assume they will use a shared integer to pass a sequence of integer values.

Define the shared integer to be named seq, and start the integer with an initial value of -1,

as in the declaration:

int32 seq = -1;

Perform each step, and answer the corresponding question before moving on.

As in previous labs, remove the items in main.c that run the Xinu shell.

**A. Start with uncoordinated processes.**

Write the code for a producer process that generates a sequence as follows:

1. Generate a sequence of integers (all the even integers from 0 through 800),

2. Place each successive value in the shared variable, seq

3. Exit

Write the code for a consumer process that accepts the sequence:

1. Repeatedly accept the next item in the sequence, check that the item has the value

expected (i.e., receive all the values in the correct order). Report an error if a

value is incorrect.

2. Exit

Modify main.c by inserting code to:

1. Create and resume two processes of priority 10, one to execute your producer and

one to execute your consumer

2. Sleep for 1 second

3. Print the value in seq.

4. Exit

**Answer the question below** and **then** perform the additional steps below to test various

ways to coordinate the producer and consumer processes (each step requires modifying

main).

**B. Use a single semaphore, mutex, for mutual exclusion.**

Have both the producer and consumer surround each reference to the shared variable with

wait(mutex) and signal(mutex). As before, have the main process sleep for a

second after creating and resuming the two processes, and then print the final value in the

shared variable. Test the results.

**C. Repeat the previous step, but give the producer priority 10 and the**

**consumer priority 12. Test the results.**

**D. Repeat the previous step, but reverse the priorities so the producer has a**

**higher priority than the consumer. Test the results.**

**E. Use two semaphores to coordinate the two processes in a traditional**

**producer-consumer configuration. Test the results.**

**Questions**

Place **short answers** to the following questions in a file named questions.pdf. In the

case of using document software (e.g., Word), please make sure to convert the file to

**PDF**.

Explain what happened in Step A (i.e., tell which processes ran and the order in

which they ran to produce the observed outcome.

Explain what happened in Step B. (i.e., tell the order in which processes ran and

why the consumer did or did not receive all values in the sequence).

Explain what happened in Step C (i.e., tell the order in which processes ran and

why the consumer did or did not receive all values in the sequence).

Explain what happened in Step D. (i.e., tell the order in which processes ran and

why the consumer did or did not receive all values in the sequence).

Explain what happened in Step E. (i.e., tell the order in which processes ran and

why the consumer did or did not receive all values in the sequence).

**Submission**

General instructions:When implementing code in the labs, please maintain separate versions/copies of code so

that mistakes such as unintentional overwriting or deletion of code is prevented. This is in

addition to the efficiency that such organization provides. You may use any number of

version control systems such as GIT and RCS. Please make sure that your code is

protected from public access. For example, when using GIT, use git that manages code

locally instead of its on-line counterpart github. If you prefer not to use version control

tools, you may just use manual copy to keep track of different versions required for

development and testing. More vigilance and discipline may be required when doing so.

You must submit a directory named lab3that contains both required files:

●

●

Your final version of main.c from Xinu

questions.pdf, containing your answers

Go to the directory where lab3 is a subdirectory.

For example, if /homes/alice/cs354/lab3 is your directory structure, go to

/homes/alice/cs354.

Type the following command to submit the directory with turnin:

turnin -c cs354 -p lab3 lab3

Be sure the files inside the directory are named exactly main.c and

questions.pdf.

You can check/list the submitted files using

turnin -c cs354 -p lab3 -v