CS 503 - Fall 2014

Lab 1
Process Management using Semaphores
The Farmer and The Vegetarians

Due Sunday, September 14th, 2014 at 11:59 PM

Objectives:

By the end of this lab students will be able to:

- Understand how to create processes in XINU and pass parameters to them
- Create multiple processes that all need to access a shared buffer
- Perform process synchronization using semaphores to provide mutual exclusion to a shared resource

Background

Suppose there exists a field where farmers grow carrots. There also exists one or more hungry vegetarians who want to buy and consume the carrots that the farmers produce and then go away to do some other business before getting hungry again and stop by the field to buy more carrots. The farmers can plant one carrot at a time in the field and it takes a certain amount of time for each carrot to grow and be ready for sale. The field also has a fixed size and can only hold a fixed number of carrots.

Your job is to simulate a farm that produces and sells carrots. Farmers work to raise carrots, and each farmer has a rate of production. In the simulation, a farmer will be a process that repeatedly sleeps for and then produces a carrot. The sleep time is chosen for each farmer to set the farmer's rate. Vegetarians periodically come to the farm to buy carrots. In the simulation, each vegetarian is represented by a process that repeatedly sleeps and consumes carrots. A vegetarian process has two parameters:

- a sleep time (which determines how often the vegetarian comes to the farm)
- a hunger value that specifies how many carrots the vegetarian buys on a given visit.

Farmers get paid by how many carrots they sell. So, a farmer tags each carrot with a 1-character label. You can use 'A' for farmer 1, 'B' for farmer 2, and 'C' for farmer 3. When a vegetarian buys a carrot, your system must keep a record of which farmer produced the carrot.

Hint: you may want to use a circular character buffer.

For the initial lab, assign all processes the same priority (20). We will supply a set of sleep times and hunger values.

A master process controls the simulation and prints results. The master starts the farmer and vegetarian processes, sleeps for 30 seconds, and then generates a report as shown below.

Hint: because process creation is relatively expensive, you may want to find way to start all processes at the same time (think about signaln).

The report looks like this:

```
Farmer A: sold xxxxx carrots
Farmer B: sold xxxxx carrots
Farmer C: sold xxxxx carrots

Vegetarian a: XXX carrots from farmer A, XXX from farmer B, and XXX from farmer C
Vegetarian b: XXX carrots from farmer A, XXX from farmer B, and XXX from farmer C
Vegetarian c: XXX carrots from farmer A, XXX from farmer B, and XXX from farmer C
```

Setup

In /homes/cs503/xinu there is a file called xinu-fall2014-lab1.tar.gz that contains a start to the code for this simulation. Unpack:

```
        tar zxvf /u/u3/cs503/xinu/xinu-fall2014-lab1.tar.gz
```

This will create a directory called xinu-fall2014-lab1.

Along with the main code for XINU, this tarball contains the following files (additional explanation of the contents of the files is in the following sections).

- system/farm.c - function declarations for the farm simulation code
- include/farm.h - parameter declarations for the farm simulation
- system/main.c - the main XINU process

What's in system/farm.c

The file farm.c file contains the function declarations for 3 functions that you must implement:

- void start_farm(void) - the initialization function. This creates all necessary processes and starts the farm simulation running.
- void stop_farm(void) - stops the simulation.
- void print_farm_report(void) - prints the farm report as described above.

You are required implement the bodies of these functions. Any additional variables that need to be created for your simulation should be added to farm.c as needed.

What's in include/farm.h

The farm.h file contains the parameters for the simulation. Feel free to change these parameters during testing of your code, but do not place any variables specific to your implementation in farm.h. As part of grading the teaching assistants will replace farm.h for each test case.

Values defined in farm.h:

- NFARMERS - the number of farmer processes. You can assume this number will always be greater than zero.
- NVEGETARIANS - the number of vegetarian processes. You can assume this number will always be greater than zero.
- FIELDSIZE - the size of the field. You can assume this number will always be greater than zero.
- farmer_tags - an array of tags for the farmer processes. The tag for the first farmer is farmer_tags[0]. The tag for the second farmer is farmer_tags[1], etc. This array will always be NFARMERS in size.
- farmer_grow_times - an array of values representing the time it takes for each farmer to produce a carrot. The grow time for the first farmer is farmer_grow_times[0]. The grow time for the second farmer is farmer_grow_times[1], etc. This array will always be NFARMERS in size.
- vegetarian_tags - an array of tags for the vegetarian processes. The tag for the first vegetarian is vegetarian_tags[0]. The tag for the second vegetarian is vegetarian_tags[1], etc. This array will always be NVEGETARIANS in size.
- vegetarian_sleep_times - an array of values representing the time each vegetarian sleeps before buying more carrots. The sleep time for the first vegetarian is vegetarian_sleep_times[0]. The sleep time for the second vegetarian is vegetarian_sleep_times[1], etc. This array will always be NVEGETARIANS in size.
- vegetarian_hungers - an array of values representing the number of carrots each vegetarian buys before returning to sleep. The hunger value for the first vegetarian is vegetarian_hungers[0]. The hunger value for the second vegetarian is vegetarian_hungers[1], etc. This array will always be NVEGETARIANS in size.

What's in system/main.c

The main.c file contains the code for the main process that will call the functions to execute the farm simulation. The functions will always be called in the same order:

1. start_farm
2. sleep for some amount of time
3. stop_farm
4. print_farm_report

The teaching assistants may change the main.c file during testing so do not put any code specific to your implementation in main.c.

Additional Requirements:

- The field has fixed size of FIELDSIZE, so a farmer cannot grow more carrots if the field is full. If this happens the farmer must wait until there is an open spot in the field. Farmers cannot share spots in the field.
- Vegetarians cannot share carrots. If there is only 1 carrot available in a field and 2 vegetarians want to buy and consume it, only one should be allowed to do so.

Extra credit

Experiment with process priorities by making

1. The farmers have higher priority than the vegetarians
2. The vegetarians have higher priority than the farmers
3. Each vegetarian to have a unique priority (high, medium, low)

Compare the results. Which approach results in better "fairness"?

What to turn in

Submit using turnin command your complete source code (all of XINU) including the any files you added to complete the lab. In the 'system' directory include a PDF file with a write up discussing:

- The details behind your implementation. As part of this discussion write answers to the following questions:
  - How does your solution guarantee that only one process will attempt to remove a carrot at a given time?
  - In your solution, when a producer needs to insert a carrot in the queue, are consumers also excluded, or can consumers continue concurrently?
  - When a consumer starts to extract carrots, does your system guarantee that the consumer will receive contiguous locations in the queue, or do consumers contend for carrots?
- The steps you took to complete the requirements
- Your results from the extra credit experiment

To turn in your lab use the following command

```
turnin -c cs503 -p lab1 xinu-fall2014-lab1
```

assuming xinu-fall2014-lab1 is the name of the directory containing your code.

If you wish to, you can verify your submission by typing the following command:

```
turnin -v -c cs503 -p lab1
```

Do not forget the -v above, as otherwise your earlier submission will be erased (it is overwritten by a blank submission).

Note that resubmitting overwrites any earlier submission and erases any record of the date/time of any such earlier submission.

We will check that the submission timestamp is before the due date; we will not accept your submission if its timestamp is after the due date. Do NOT submit after 11:59 PM Eastern Time.