National University of Singapore
School of Computing

CS2106 # Laboratory 5 Semester 1 11/12

Deadline

30 September, 2011, Friday, 10:00pm

Platform

This exercise can be done on either a Linux PC or SunFire.

Submission

Your working directory should be named `lab05-A000000X` where `A000000X` is your matriculation number. Create a tar ball named `lab05-A000000X.tar` containing your version of the source code by:

```
tar cvf lab05-A000000X.tar lab05-A000000X/*
```

and submit the tar file into the IVLE workbin named Lab 5.

Marking Scheme

This lab exercise is worth 10 marks. 0.1 marks will be deducted per minute after the deadline for late submission, and 3 marks will be deducted if the tar file or the subdirectory is not named properly with the right matriculation number.

You will implement a simulation of the Dining Savages problem (Tutorial 4 Question 3) using only pthread APIs. A solution to the Dining Savages problem using semaphores has been given to you. Download the tar ball from `http://www.comp.nus.edu.sg/~cs2106/lab05-A000000X.tar` and untar the downloaded tar ball using:

```
tar xvf lab05-A000000X.tar
```

A subdirectory `lab05-A000000X` is now created. Rename the directory by replacing the string "A000000X" with your matriculation number. Inside the directory are a `Makefile` and the semaphore-based solution, `savages.c`. You can compile the code using the `make` command.

Skeleton Code

Read `savages.c` and understand the code provided. If you encounter any unfamiliar function call, use the man pages to find out what it is for.

The program simulates N savages and one cook, and it takes three command line arguments, N, the size of the tribe, M, the size of the pot, and k, the number of times a savage eats.

N savages eat from a pot that can hold M servings of food. When a savage wants to eat, he helps himself from the pot with a single serving, unless it is empty. If the pot is empty, the savage wakes up the cook and then waits until the cook has refilled the pot. When woken up, the cook refills the pot with M servings of food, and goes back to sleep. A savage repeatedly eats k times, while the cook cooks $\lceil \frac{Nk}{M} \rceil$ times.

At the end of the simulation, the number of food servings left in the pot is

$$(M - Nk \bmod M) \bmod M.$$

Your Tasks

Your tasks in this exercise are to change the implementation of the savage processes and the cook process from using semaphore to using pthread library, so that the behaviour of the processes is the same as that described in the Dining Savage problem.

Your implementation must avoid deadlock and synchronize the processes properly so that, at the end of the simulation, the amount of left over in the pot is always correct.

Note that you are not allowed to force the threads to run one after another (e.g., Savage 1 eats k times, then Savage 2 eats k times, etc.).

Further, do note that `pthread_cond_wait` must be called with mutex locked by *the calling thread*. Otherwise, undefined behavior may occur.

Example

Assuming your executable is called `savages`, if you run:

```
savages 7 11 100
```

then you are simulating 7 savages eating from a pot of size 11, with each savage eating 100 times. The program should output:

```
4
```

because that is the amount of left over in the pot after the simulation. Note that a buggy program may still output a correct answer.

THE END