

Concurrent Systems

Assignment 1

1 Introduction

The course is about concurrency. It is your solutions to the concurrency problems that we want to see, the rest of your code should be as simple as possible.

2 Your Tasks

2.1 The Sausage Sizzle

To raise funds for the new school hall, students are selling sausages in front of a heavily frequented warehouse. They have set up two barbecues, which produce sausages on a regular basis. These are then sold to the customers as they come in.

Write a simulation of this problem, using the synchronized keyword as the synchronisation mechanism. Your solution should print out the details of what is happening, such as:

- Barbecue 2 has another sausage ready.
- Customer 42 buys 2 sausages.

Your solution should generate 100 threads, each of which requests up to 3 sausages. Once there are no requests left, your program should terminate (even if there are sausages left over).

2.2 The Harbour Shuttle

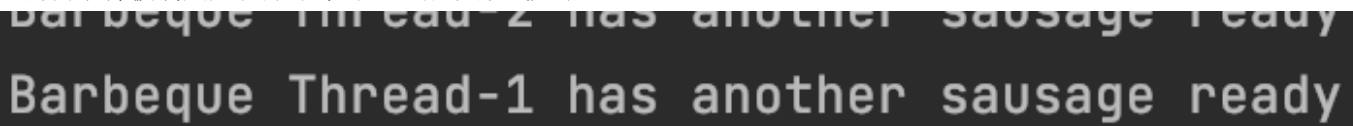
To help easing traffic problems in Auckland, a company has introduced a shuttle service across the harbour. The shuttle seats 10 people and operates on a demand basis: once all 10 seats are taken, the shuttle crosses the harbour and waits there until 10 people on the other side want to cross the harbour.

Write a simulation of this problem using semaphores. Your solution should print out the details of what is happening, such as:

- Person 3 is boarding the shuttle on the North Shore.
- Shuttle is going from the North Shore to Auckland.
- Person 4 is leaving the shuttle in Auckland.

Your solution should generate 100 threads, simulating people evenly distributed at both sides of the harbour (Auckland and the North Shore). Once all people have travelled from one side of the harbour to the other, your program should terminate.

两个小作业均需创建100个客户线程和1-2个的服务线程，用控制台程序print输出每个线程的几步动作直至100个客户的需求都被满足后程序结束即可，展现形式如下：



```
Barbeque Thread-1 has another sausage ready
```

作业2.1的关键是using synchronised, 进一步解释：The assignment brief does state "Your solution should generate 100 threads, each of which requests up to 3 sausages.". The 100 threads are customers, each of whom are racing to get at some amount of cooked sausages. We know there's 2 BBQs available. You could certainly model the BBQs as a thread each if you wish. The main goal of this task appears to be use of the synchronized keyword. If you've got 100 threads racing to use a synchronized number, or numbers, then my read is you're on the right track. It makes indeed sense to also implement the 2 BBQs as threads. These are the producers (hint), while the customer threads are the consumers. You have 100 customer threads, each requesting up to 3 sausages. In addition to that, there are two BBQ threads, each producing sausages until they are interrupted. The customer threads are the consumers and the BBQ threads are the producers. They use one common count/tray. You can use join() in the main program to wait for the customers threads to terminate, and once that has happened, to interrupt the BBQ threads. This might result in leftover sausages, but this is allowed according to the assignment specification (and probably what happens in the real world). Print statements will need to be within synchronised blocks along with the code actually modifying shared resources.

作业2.2的关键是using semaphores, 进一步解释：The 50 people on each side try to get seats on the shuttle, but only 10 will be successful for each crossing. This is where the semaphores come in. A semaphore with a value of 10 means that 10 acquire() are successful; the other threads are blocked until the semaphore becomes available again. When entering the shuttle, a thread tries to acquire the semaphore that controls entering the shuttle. Once that has been successful, the crossing begins until the thread is able to acquire the semaphore that controls leaving the shuttle. The shuttle releases the semaphores at the appropriate times. You can operate the shuttle iteratively until it is not needed any more. To find out whether the shuttle is still needed, you can use the same method that I suggested for the sausage sizzle problem: use join() for all the people threads in a loop, and once the loop has finished, you know that there are no people left who require travel, so you can interrupt the shuttle thread.