Course 'Operating Systems Architecture' – tutorial: Condition variables

UFAZ, L2
Lecturer: Pierre Parrend, Rabih Amhaz
This complementary material is made available by Remzi H. Arpaci-Dusseau and Andrea C. Arpaci-Dusseau on:
http://pages.cs.wisc.edu/~remzi/OSTEP/
http://pages.cs.wisc.edu/~remzi/OSTEP/Homework/homework.html

This homework lets you explore some real code that uses locks and condition variables to implement various forms of the producer/consumer queue discussed in the chapter. You'll look at the real code, run it in various configurations, and use it to learn about what works and what doesn't, as well as other intricacies. Read the `README` for details.

1. Exercise 1

Our first question focuses on `main-two-cvs-while.c` (the working solution).
First, study the code.

1.1 Do you think you have an understanding of what should happen when you run the program?

2. Exercise 2

Run with one producer and one consumer, and have the producer produce a few values. Start with a buffer (size 1), and then increase it.

2.1 How does the behaviour of the code change with larger buffers? (or does it?)

2.2 What would you predict `num_full` to be with different buffer sizes (e.g., `-m 10`) and different numbers of produced items (e.g., `-l 100`), when you change the consumer sleep string from default (no sleep) to `-C 0,0,0,0,0,0,1`?

3. Exercise 3

If possible, run the code on different systems (e.g., a Mac and Linux).

3.1 Do you see different behaviour across these systems?

4. Exercise 4

Let's look at some timings.

4.1 How long do you think the following execution, with one producer, three consumers, a single-entry shared buffer, and each consumer pausing at point `c3` for a second, will take?

```
./main-two-cvs-while -p 1 -c 3 -m 1 -C 0,0,0,1,0,0,0:0,0,0,1,0,0,0:0,0,0,1,0,0,0
-l 10 -v -t
```

5. Exercise 5

Now change the size of the shared buffer to 3 (`-m 3`).

5.1 Will this make any difference in the total time?

6. Exercise 6

Now change the location of the sleep to `c6` (this models a consumer taking something off the queue and then doing something with it), again using a single-entry buffer.

6.1 What time do you predict in this case?

```
./main-two-cvs-while -p 1 -c 3 -m 1 -C 0,0,0,0,0,0,1:0,0,0,0,0,0,1:0,0,0,0,0,0,1
-l 10 -v -t
```

7. Exercise 7

Finally, change the buffer size to 3 again (`-m 3`).

7.1 What time do you predict now?

8. Exercise 8

Now let's look at `main-one-cv-while.c`.

8.1 Can you configure a sleep string, assuming a single producer, one consumer, and a buffer of size 1, to cause a problem with this code?

9. Exercise 9

Now change the number of consumers to two.

9.1 Can you construct sleep strings for the producer and the consumers so as to cause a problem in the code?

10. Exercise 10

Now examine `main-two-cvs-if.c`.

10.1 Can you cause a problem to happen in this code? Again consider the case where there is only one consumer, and then the case where there is more than one.

11. Exercise 11

Finally, examine `main-two-cvs-while-extra-unlock.c`.

11.1 What problem arises when you release the lock before doing a put or a get?

11.2 Can you reliably cause such a problem to happen, given the sleep strings?

11.3 What bad thing can happen?