5condition-variables-lab

CST 334: Operating Systems
Dr. Glenn Bruns

Lab: condition variables

In this lab we will create a simple synchronized buffer. This buffer holds at most a single value. A thread can't write to the buffer unless it's empty, and can't read from the buffer unless it holds a value.

The state of the buffer is captured with two variables: buf, and count. The 'buf' variable holds the single value that can be stored in the buffer. The 'count' variable is 0 or 1. It is 0 if the buffer is empty, and 1 is the buffer is non-empty.

Check out the pthreads cheat sheet for a summary of the pthreads API.

1. On mlc104, copy the file

```
/home/CLASSES/brunsglenn/cst334/labs/cond-vars/sync_buf_skeleton.c
```

to a directory of your own, and rename it to sync_buf.c. This code is similar to the code we saw in lecture, but there is no buffer object.

2. Read and understand the code. Remember: the reader will read from the buffer, and the writer will write to the buffer. Compile your code with

```
$ gcc -pthread -lm -o sync_buf sync_buf.c
```

3. Now you will fill in the "holes" in the file identified by "your code here". First take care of the holes in method reader(). Does the reader ever need to wait for the writer? Why? Does the reader ever need to notify the writer? Why? Are lock/unlock operations needed?

See hints below -- but try hard before referring to the hints.

4. Fill in the "holes" in method writer().

5. How can you be sure that you actually did implement a synchronized buffer? Do the existing print statements in the code give you enough information? Add print statements if you think they are needed, and run your code again.

6. If you still have time, try solving the problem with a single condition variable, instead of two.

Hints:

1. -

2. -

3. The reader can only read the buffer when something is in the buffer, so sometimes it will need to wait for the writer. On the other hand, the writer must wait for the buffer to be empty before writing, so the reader must let the writer know after it has emptied the buffer.

The lock must always be held when wait or signal calls are made.

Read on only if you need a better hint…

Pseudocode for the reader:

```
grab the lock
while (buffer empty) {
    wait on the condition variable
}
read the value in the buffer
make buffer as empty
signal that the buffer is empty
release the lock
```

4. The writer must wait for the buffer to be empty before writing, and must notify the reader when it has put something in the buffer.

5. -