6cond-vars-dahlin-lab

CST 334: Operating Systems
Dr. Glenn Bruns

Lab: Anderson/Dahlin method

In this lab we will continue work on our synchronized buffer code by turning the synchronized buffer into a shared object, following the Anderson/Dahlin method. The file needed for the lab can be found on the hosting machine at:

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

Copy this file to a file named `sync_buf_obj.c` in a directory of your own, and make sure you can compile it:

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

Following the Anderson/Dahlin method, fill the “holes” in the code that are indicated by comments with “YOUR CODE HERE”.

1. Look at the code to make sure you understand how it is organized. We are modeling a one-element buffer.

2. Add code in sbuf_create, the constructor of ‘sbuf’ objects, to initialize the synchronization and state variables. Which synchronization variables need to be initialized? Which state variables need to be initialized?

3. Add code in sbuf_read, the “method” for reading from the buffer.

4. Add code in sbuf_write, the method for writing to the buffer.

5. Figure out how you can see if your code is working right. For example, do you need to add print statements to the code to help you debug it?

6. You may also want to write an awk script or other code to look at the output of the code (in particular, the output produced by the print statements you add) to test if the output is correct.

If you still have time, modify the code so that it supports at least three writing threads and three reading threads. Does your code still work? If not, fix it. You may need to modify the state and synchronization variables.

REMINDER: Here are the high-level steps of the Anderson/Dahlin method:

- start with normal class design
- add a lock to the class; enclose method bodies with lock/unlock calls
- add 0 or more condition variables to the class
- add wait calls within loops
- add signal and broadcast calls

Hints:

1. -

2. Synchronization variables always need to be initialized. For this problem, state variable buf does not need to be initialized because initially count is 0.

3. Remember that in the Dahlin method, the first thing to do is enclose the method within lock/unlock operations. The sbuf_read method can only read the buffer when something is in the buffer, so sometimes it will need to wait for another thread. Other threads may wait for a value to be read from the buffer, so sub_read must let other threads know when it reads a value out of the buffer.

4. Does sbuf_write ever need to wait on another thread before writing? Does sbuf_write ever need to notify another thread after it has written?

5. -

6. -