EECS 111 - System Software

Spring 2010

Assignment 4

Posted: Wednesday, April 28, 2010

Due: Tuesday, May 11, 2010, 12pm (noon)

A. Discussion: Process Synchronization

The goal of these exercises is to review and clarify the understanding of essential aspects covered in recent lectures.

- Synchronization primitives: Exercises 6.26, 6.28, and 6.33.
- Application problems: Exercise 6.37 and 6.38

These topics are planned to be discussed in this week's discussion session.

B. Project: Producer Consumer Problem, Mutex, Condition Variables

The goal of this project assignment is to improve the bounded buffer implementation in our producer consumer example by use of Pthread mutex and condition variables. Inter-thread communication through proper synchronization primitives will greatly benefit the efficiency of the bounded buffer in terms of space and time.

As before, we will use the EECS department servers which offer parallel execution of threads on multiple CPUs.

Step 1: Setup

- We will use the very same setup as in the previous project. Please refer to the instructions of Assignment 3 for details.
- For this project, we will extend the code written previously. To get started, create a copy of your previous source code and name it **`prodcons2.c`**. (You may want to start with the posted solution for Assignment 3 if your own code has had any problems.)
- For this assignment, we will only modify the implementation of the bounded buffer. All other parts in your program should remain unmodified. In particular, do *not* modify any of the following functions:

```
int fibonacci(int n);
void *produce(void *arg);
void *consume(void *arg);
int main(int argc, char **argv);
```

Step 2: Fully utilize the buffer size

In the previous assignment, we have been using the initial bounded buffer implementation from chapter 3 in the textbook (version 1). As outlined in the book and discussed in Lecture 9, this implementation cannot fully utilize the allocated buffer space. At most **`BUFFER_SIZE-1`** items can be stored.

In chapter 6, the textbook extends the implementation by an additional variable **`counter`** in order to fix the space inefficiency. Extend and modify your code the same way. This should result in the bounded buffer implementation version 2 (see Lecture 9).

Step 3: Implement proper synchronization

As also discussed in the textbook and in Lecture 9, version 2 of the bounded buffer contains a race condition that renders this implementation unsafe! This needs to be fixed now.

To do this, we will use Pthread mutex and condition variables. Add the required synchronization variables to the shared variables of the buffer and use them by appropriate pthread API calls in the **`send()`** and **`receive()`** functions so that both the race condition and busy waiting in the bounded buffer are eliminated.

Note that there is not much code to change in your program.

Step 4: Compile and test your implementation

When running your improved program, your execution log should look very similar to the previous assignment. However, some changes may be noticeable. Briefly list the *possible changes* in the log and describe why (or why not) you notice these in your actual execution.

Again, state your expectation of the execution times (user, system, and elapsed/real time), analyze your program using the **`/usr/bin/time`** command for **`max=42`** (note the server name you are using), compute the new CPU load, and compare the results to the previous assignment. Briefly explain why or why not the observed results match your expectation.

Deliverables:

1. Statement: “I have read the Section on Academic Honesty in the UCI Catalogue of Classes (available online at http://www.editor.uci.edu/catalogue/appx/appx.2.htm#gen0) and submit this work accordingly.”

2. Source file **`prodcons2.c`**, execution log **`prodcons2.log`**, and a brief text file **`prodcons2.txt`** explaining the results and answering the above questions (3-4 paragraphs) [100 points].

Submission instructions:

To submit your homework, send the deliverables in an email with subject “EECS111 HW3” to the course instructor at doemer@uci.edu.

To ensure proper credit, be sure to send your email before the deadline:
Tuesday, May 11, 2010, at 12:00pm (noon).

--

Rainer Doemer (EH 3217, x4-9007, doemer@uci.edu)