CSc 332 (L) - Operating Systems

Lab – Spring 2018

Task 6: Cigarette Smokers Problem

Due May 2 2018

Problem Description

Consider a system with 3 smoker processes and 1 agent process. Each smoker continuously rolls a cigarette and then smokes it. The smoker needs three ingredients: tobacco, paper, and matches. One of the smokers has paper, another has tobacco, and the third has matches. The agent has an infinite supply of all three materials and (randomly) places two of the ingredients on the table each time. The smoker who has the remaining ingredient then makes and smokes a cigarette, signaling the agent on completion. The agent then puts out another two of the three ingredients, and the cycle repeats.

TO DO: Write a program to synchronize the agent and smoker processes either using traditional UNIX semaphores or using the pthread library and pthread_mutex.

Instructions

- You need to use the `sem.h` header file in your semaphore-based solution.
- If you choose to use the traditional semaphore technique, you will want to follow the semaphore code in ProducerConsumer.c and the associated sem.h header file.
- Although the problem description gives the agent an infinite supply of ingredients, you may set a reasonable finite number of iterations.
- You need to include an explanation of how your code succeeds in synchronizing the agent and the smokers and avoiding starvation as well as interleaving and race conditions. Provide output showing the sequence of events for several iterations.
- Extra credit: I will explain in more detail. See Hollingworth's pseudocode below. Instead of having the appropriate smoker wakened by the agent, let's suppose that the agent doesn't know which smoker has what ingredient, and let's suppose that the smokers don't know what the agent has placed in the Critical Sector without looking. Thus, implement a queue of smokers each of whom, when she accesses the CS, checks to see if the ingredients match what she has, and if so, takes them and smokes, and if not, awakens the next agent in the queue and then sleeps. The next smoker then checks the ingredients in the CS and behaves the same way. Each smoker gets on the back of the queue when she finishes smoking. It is not technically necessary to remove the ingredients placed in the CS as they only match one smoker.

Zip your source & explanation text files into a single folder as: task6_*lastname*.zip. Email your zip file with the subject line, "Task 6 – CSc332G- *lastname*".

Cigarette Smoker's Problem as presented by Prof. Jeffrey Hollingsworth, University of Maryland

Problem

There are four processes in this problem: three smoker processes and an agent process. Each of the smoker processes will make a cigarette and smoke it. To make a cigarette requires tobacco, paper, and matches. Each smoker process has one of the three items. I.e., one process has tobacco, another has paper, and a third has matches. The agent has an infinite supply of all three. The agent places two of the three items on the table, and the smoker that has the third item makes the cigarette. Synchronize the processes.

Please note this is the formulation of the problem used in recent OS texbooks, but not the problem as originally described by Parnas in CACM March 1975.

Solution

This seems like a fairly easy solution. The three smoker processes will make a cigarette and smoke it. If they can't make a cigarette, then they will go to sleep. The agent process will place two items on the table, and wake up the appropriate smoker, and then go to sleep. All semaphores except lock are initialized to 0. lock is initialized to 1, and is a mutex variable.

Here's the code for the agent process.

```
1   do forever {
2     P( lock );
3     randNum = rand( 1, 3 ); // Pick a random number from 1-3
4     if ( randNum == 1 ) {
5       // Put tobacco on table
6       // Put paper on table
7       V( smoker_match );  // Wake up smoker with match
8     } else if ( randNum == 2 ) {
9       // Put tobacco on table
10      // Put match on table
11      V( smoker_paper );  // Wake up smoker with paper
12    } else {
13      // Put match on table
14      // Put paper on table
15      V( smoker_tobacco ); } // Wake up smoker with tobacco
16    V( lock );
17    P( agent );  //  Agent sleeps
18    }  // end forever loop
```

I will give code to one of the smokers. The others are analogous.

```
1   do forever {
2     P( smoker_tobacco );  // Sleep right away
3     P( lock );
4     // Pick up match
5     // Pick up paper
6     V( agent );
7     V( lock );
8     // Smoke (but don't inhale).
9   }
```

The smoker immediately sleeps. When the agent puts the two items on the table, then the agent will wake up the appropriate smoker. The smoker will then grab the items, and wake the agent. While the smoker is smoking, the agent can place two items on the table, and wake a different smoker (if the items placed aren't the same). The agent sleeps immediately after placing the items out. This is something like the producer-consumer problem except the producer can only produce 1 item (although a choice of 3 kinds of items) at a time.