Project 7—Producer–Consumer Problem

Submissions: your code and the output (screenshots)

In this project, you will use Pthreads to implement a solution to the producer and consumer problem. For this project, you will use standard counting semaphores for *empty* and *full*, which count the number of empty and full slots in the buffer, and a mutex lock, which protects the actual insertion or removal of items in the buffer. The producer and consumer—running as separate threads—will move items to and from a buffer that is synchronized with the *empty*, *full*, and mutex structures.

The Buffer

Internally, the buffer will consist of a fixed-size array of type buffer_item (which will be defined using a typedef). The array of buffer_item objects will be manipulated as a circular queue. The definition of buffer_item, along with the size of the buffer, can be stored in a header file such as the following:

```
/* buffer.h */
typedef int buffer_item;
#define BUFFER SIZE 5
```

The buffer will be manipulated with two functions, insert_item() and remove_item(), which are called by the producer and consumer threads, respectively. A skeleton outlining these functions appears as follows.

```
#include "buffer.h"
/* the buffer */
buffer item buffer[BUFFER SIZE];
int insert item(buffer_item item) {
        /* insert item into buffer
        return 0
        */
}
int remove item(buffer_item *item) {
        /* remove an object from buffer
        placing it in item
        return 0
        */
}
```

The insert_item() and remove_item() functions will synchronize the producer and consumer. To implement the buffer, it will also require an initialization function that initializes the mutual-exclusion object mutex along with the empty and full semaphores.

The main() function will do the initializations and create separate producer and consumer threads. Once it has created the producer and consumer threads, the main() function will sleep for a period of time and, upon awakening, will terminate the application. The main() function will be passed three parameters on the command line:

1. How long to sleep before terminating
2. The number of producer threads
3. The number of consumer threads

A skeleton for this function appears as below.

```
#include "buffer.h"
int main(int argc, char *argv[]) {
        /* 1. Get command line arguments argv[1],argv[2],argv[3] */
        /*     argv[1]: sleepTime, argv[2]: producerThreads, argv[3]: comsumerThreads  */
        /*     sleepTime=atoi(argv[1]); convert command line input (a string) to an integer*/
        /* 2. Initialize semaphores and mutex */
        /* 3. Create producer thread(s) */
        /* 4. Create consumer thread(s) */
        /* 5. Sleep */
        /* 6. Exit */
}
```

The Producer and Consumer Threads

The producer thread will alternate between sleeping for a random period of time and inserting a random integer into the buffer. Random numbers will be produced using the rand() function, which produces random integers between 0 and RAND_MAX. The consumer will also sleep for a random period of time and, upon awakening, will attempt to remove an item from the buffer. An outline of the producer and consumer threads appears as below.

```
#include <stdlib.h> /* required for rand() */
#include "buffer.h"
void *producer(void *param) {
        buffer_item item;
        while (true) {
                /* sleep for a random period of time */
                sleep(...);
                /* generate a random number */
                item = rand();
                if (insert_item(item))
                        fprintf(stderr, "report error condition, Producer");
                else
                        printf("producer produced %d\n",item);
        }
}
```

```
void *consumer(void *param) {
        buffer_item item;
        while (true) {
                /* sleep for a random period of time */
                sleep(...);
                if (remove item(&item))
                        fprintf(stderr, "report error condition, Consumer");
                else
                        printf("consumer consumed %d\n",item);
        }
}
```