CSCI-3753: Operating Systems Fall 2019

Abigail Fernandes

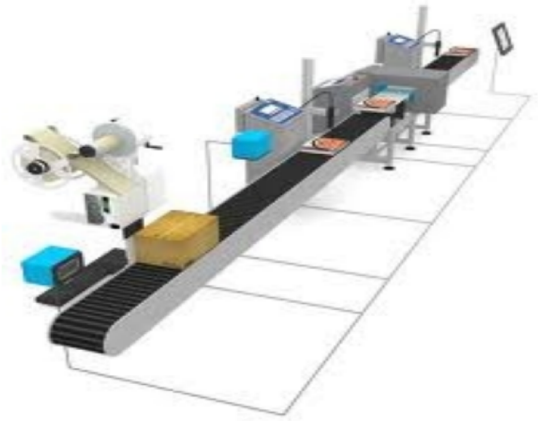
Department of Computer Science

University of Colorado Boulder

Week 8
> PA 3 Work Day

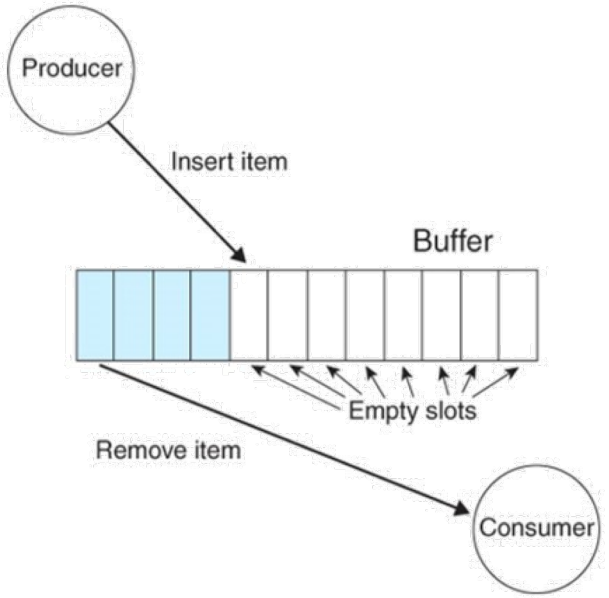
Producer Consumer Problem

- Arises when **two or more threads** communicate with each other.
- Some threads ***produce*** data while some ***consume*** data.
- Real example: Production line



Producer Consumer Problem

- Share a common, fixed-size buffer.
- Producer writes data to the buffer
- At the same time, the consumer is consuming the data (i.e., removing it from the buffer), one piece at a time.
- The **problem** is to make sure that the producer won't try to add data into the buffer if it's full and that the consumer won't try to remove data from an empty buffer.



Example of Producer – Consumer (Pthreads)

```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>

#define BUF_SIZE 100        /* Size of shared buffer */

int buffer[BUF_SIZE];   /* shared buffer */
int add = 0;            /* place to add next element */
int rem = 0;            /* place to remove next element */
int num = 0;            /* number elements in buffer */

pthread_mutex_t m = PTHREAD_MUTEX_INITIALIZER;      /* mutex lock for buffer */
pthread_cond_t c_cons = PTHREAD_COND_INITIALIZER; /* consumer waits on this cond var */
pthread_cond_t c_prod = PTHREAD_COND_INITIALIZER; /* producer waits on this cond var */

void *producer (void *param);
void *consumer (void *param);
```

Main portion of the code

```c
int main(int argc, char *argv[]) {

    pthread_t tid1, tid2;  /* thread identifiers */
    int i;

    /* create the threads; may be any number, in general */
    if(pthread_create(&tid1, NULL, producer, NULL) != 0) {
        fprintf(stderr, "Unable to create producer thread\n");
        exit(1);
    }

    if(pthread_create(&tid2, NULL, consumer, NULL) != 0) {
        fprintf(stderr, "Unable to create consumer thread\n");
        exit(1);
    }

    /* wait for created thread to exit */
    pthread_join(tid1, NULL);
    pthread_join(tid2, NULL);
    printf("Parent quiting\n");

    return 0;
}
```

Producer Thread

```c
/* Produce value(s) */
void *producer(void *param) {

    int i;
    for (i=1; i<=20; i++) {

        /* Insert into buffer */
        pthread_mutex_lock (&m);
            if (num > BUF_SIZE) {
                exit(1);  /* overflow */
            }

            while (num == BUF_SIZE) {  /* block if buffer is full */
                pthread_cond_wait (&c_prod, &m);
            }

            /* if executing here, buffer not full so add element */
            buffer[add] = i;
            add = (add+1) % BUF_SIZE;
            num++;
        pthread_mutex_unlock (&m);

        pthread_cond_signal (&c_cons);
        printf ("producer: inserted %d\n", i);
        fflush (stdout);
    }

    printf("producer quiting\n");
    fflush(stdout);
    return 0;
}
```

Consumer Thread

```c
/* Consume value(s); Note the consumer never terminates */
void *consumer(void *param) {

    int i;

    while(1) {

        pthread_mutex_lock (&m);
            if (num < 0) {
                exit(1);
            } /* underflow */

            while (num == 0) {  /* block if buffer empty */
                pthread_cond_wait (&c_cons, &m);
            }

            /* if executing here, buffer not empty so remove element */
            i = buffer[rem];
            rem = (rem+1) % BUF_SIZE;
            num--;
        pthread_mutex_unlock (&m);

        pthread_cond_signal (&c_prod);
        printf ("Consume value %d\n", i);  fflush(stdout);

    }
    return 0;
}
```