Ιωάννης Τσάμπρας 1066584 5οετης
3η εργασία Λειτουργηκά Συστήματα

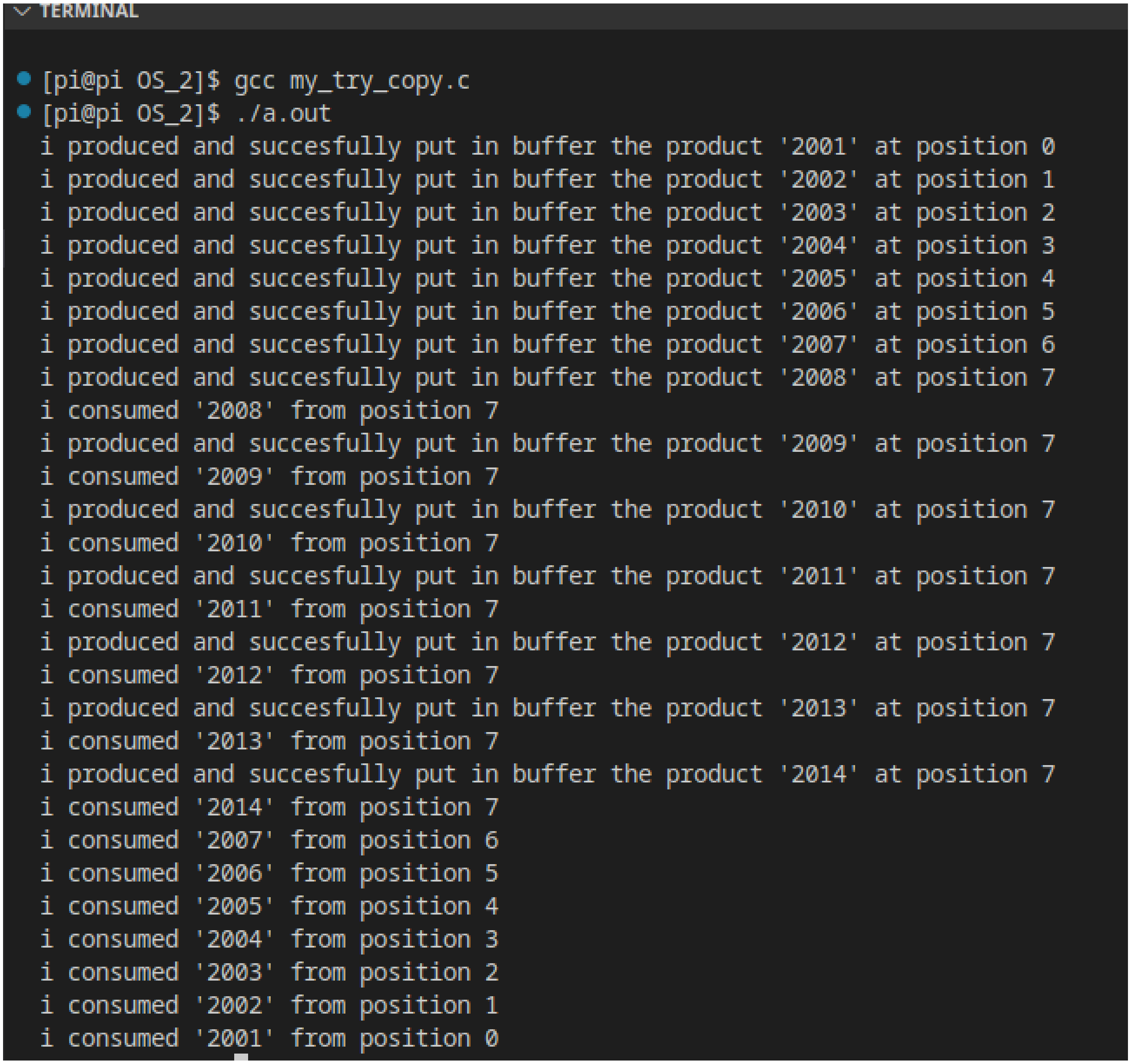
Code:

```
#include <stdio.h>
#include <semaphore.h>
#include <pthread.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>


#define SIZE 8
#define true 1
#define LIMIT 15 //to apply loop limitation

//setting thread ids arrays
int myid[100];
pthread_t tid[100];

//setting semaphore global vars
sem_t *mutex_sem;
sem_t *full_sem;
sem_t *empty_sem;


//setting shared resources gloabl vars
int buffer[SIZE];
int my_index=-1;

void *producer(void *arg){

    int product=2000;  //my product *proud farmer's face* each
time i increase the year for reasons
    int step=0; //to apply loop limitation
    while(true){
        step=step+1;
        if(step==LIMIT){
            break;
        }

        //// apply pseudocode
        sem_wait(empty_sem);
        sem_wait(mutex_sem);

        //produce
        // if there is room i put product in buffer
        if(my_index>=SIZE-1){
            printf("buffer is full!\n");
        }
        else{
            product=product+1;
            buffer[my_index]=product;
            my_index=my_index+1;
            printf("i produced and succesfully put in buffer the
product '%d' at position %d\n",product,my_index);
        }
        //
        sem_post(mutex_sem);
        sem_post(full_sem);
        //


    }


}

void *consumer(void *arg){
    int product;
    int step=0; //to apply loop limitation
    while(true){
        step=step+1;
        if(step==LIMIT){
            break;
        }


        // apply pseudocode
        sem_wait(full_sem);
        sem_wait(mutex_sem);
        //


        //consume
        if(my_index>=0){
            product=buffer[my_index-1];
            printf("i consumed '%d' from position
%d\n",product,my_index);
            my_index=my_index-1;

        }
        else{
            printf("buffer is empty!\n");
        }
        //
        //
        sem_post(mutex_sem);
        sem_post(empty_sem);
        //


    }

}



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

//setting semaphores for each shared resource
    mutex_sem= sem_open("/semaphore_mutex", O_CREAT, 0644, 1);
    sem_unlink("/semaphore_mutex");

    full_sem= sem_open("/semaphore_full", O_CREAT, 0644, 0);
    sem_unlink("/semaphore_full");

    empty_sem= sem_open("/semaphore_empty", O_CREAT, 0644,
SIZE);
    sem_unlink("/semaphore_empty");






//starting the 2 threads
    if(pthread_create(&tid[1-1], NULL, &producer, &myid[1-1]) <
0){
            printf("thread failed\n");
    }
    if(pthread_create(&tid[2-1], NULL, &consumer, &myid[2-1]) <
0){
            printf("thread failed\n");
    }
```

```
//joining the 2 threads
    pthread_join(tid[1-1], NULL);
    pthread_join(tid[2-1], NULL);


}
```

Also on github: [GitHub]()

Results:



```
● [pi@pi OS_2]$ gcc my_try_copy.c
● [pi@pi OS_2]$ ./a.out
  i produced and succesfully put in buffer the product '2001' at position 0
  i produced and succesfully put in buffer the product '2002' at position 1
  i produced and succesfully put in buffer the product '2003' at position 2
  i produced and succesfully put in buffer the product '2004' at position 3
  i produced and succesfully put in buffer the product '2005' at position 4
  i produced and succesfully put in buffer the product '2006' at position 5
  i produced and succesfully put in buffer the product '2007' at position 6
  i produced and succesfully put in buffer the product '2008' at position 7
  i consumed '2008' from position 7
  i produced and succesfully put in buffer the product '2009' at position 7
  i consumed '2009' from position 7
  i produced and succesfully put in buffer the product '2010' at position 7
  i consumed '2010' from position 7
  i produced and succesfully put in buffer the product '2011' at position 7
  i consumed '2011' from position 7
  i produced and succesfully put in buffer the product '2012' at position 7
  i consumed '2012' from position 7
  i produced and succesfully put in buffer the product '2013' at position 7
  i consumed '2013' from position 7
  i produced and succesfully put in buffer the product '2014' at position 7
  i consumed '2014' from position 7
  i consumed '2007' from position 6
  i consumed '2006' from position 5
  i consumed '2005' from position 4
  i consumed '2004' from position 3
  i consumed '2003' from position 2
  i consumed '2002' from position 1
  i consumed '2001' from position 0
```