LAB 10

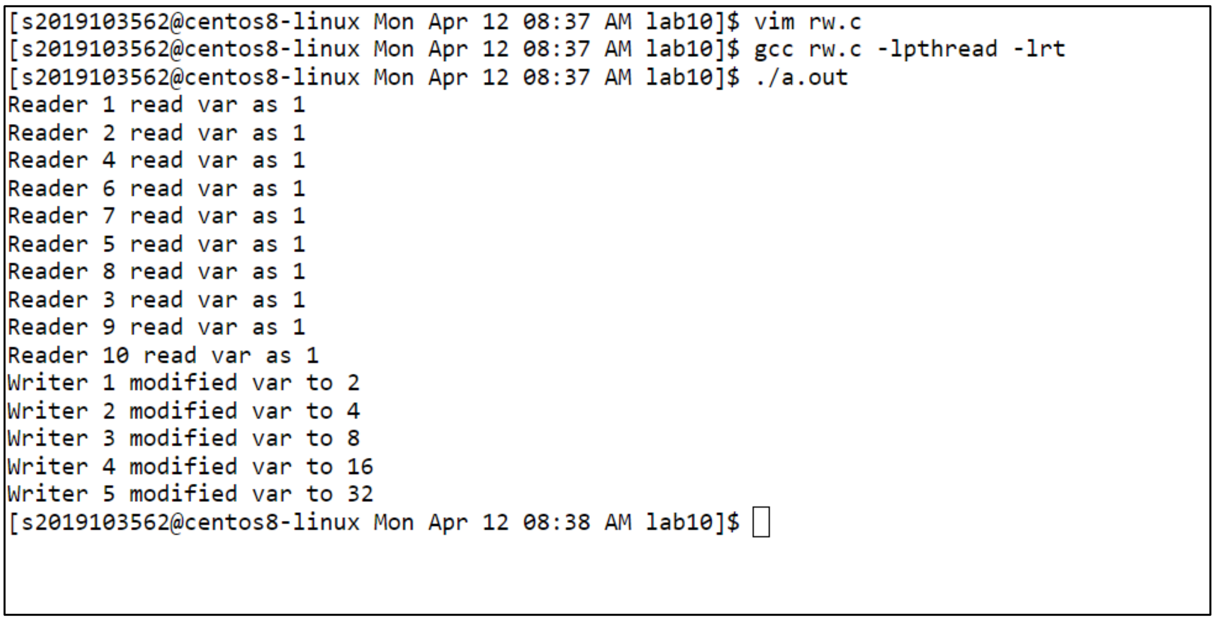
OS SYNCHRONIZATION PROBLEMS

P.PRIYADHARSHINI

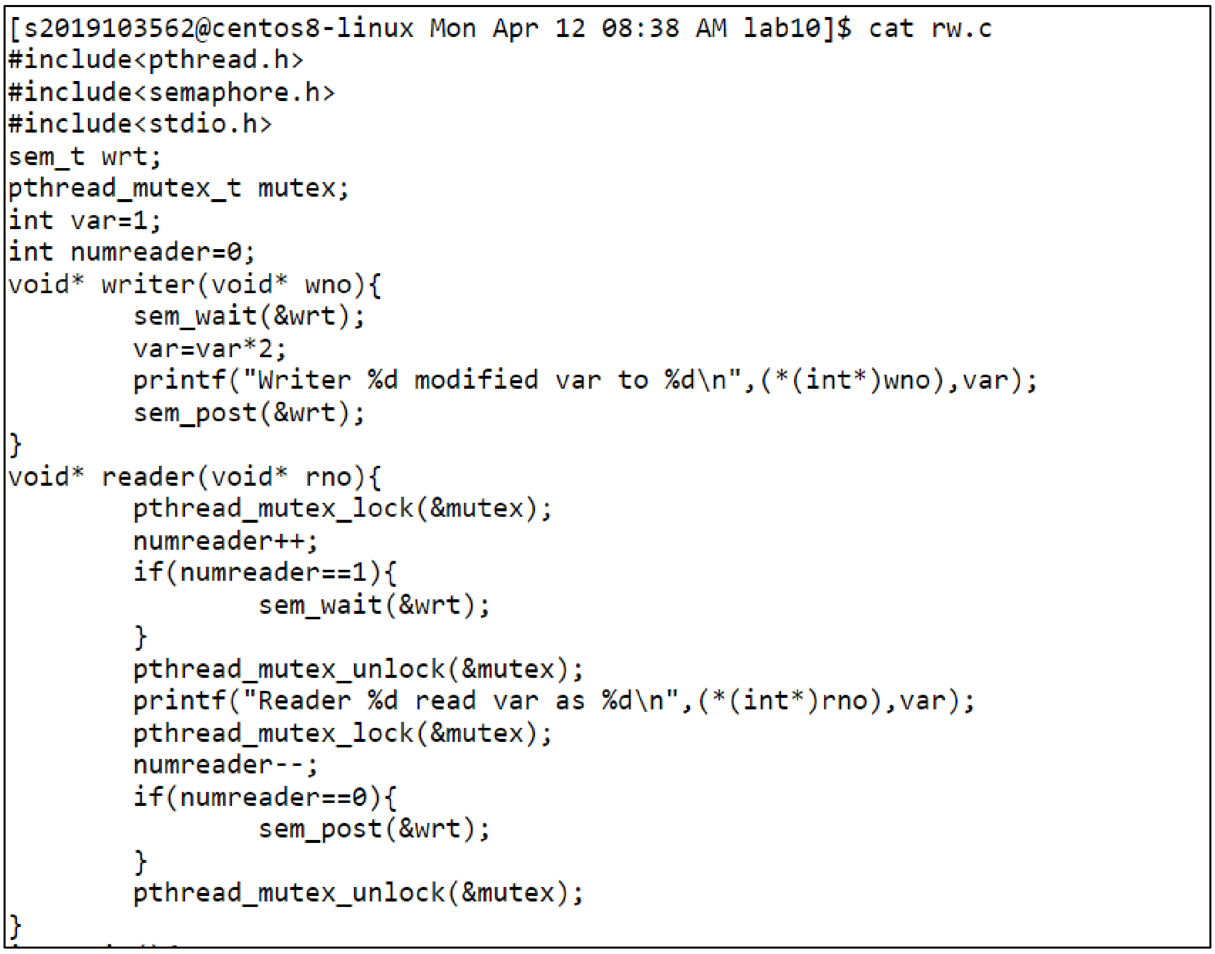
2019103562

12-04-2021

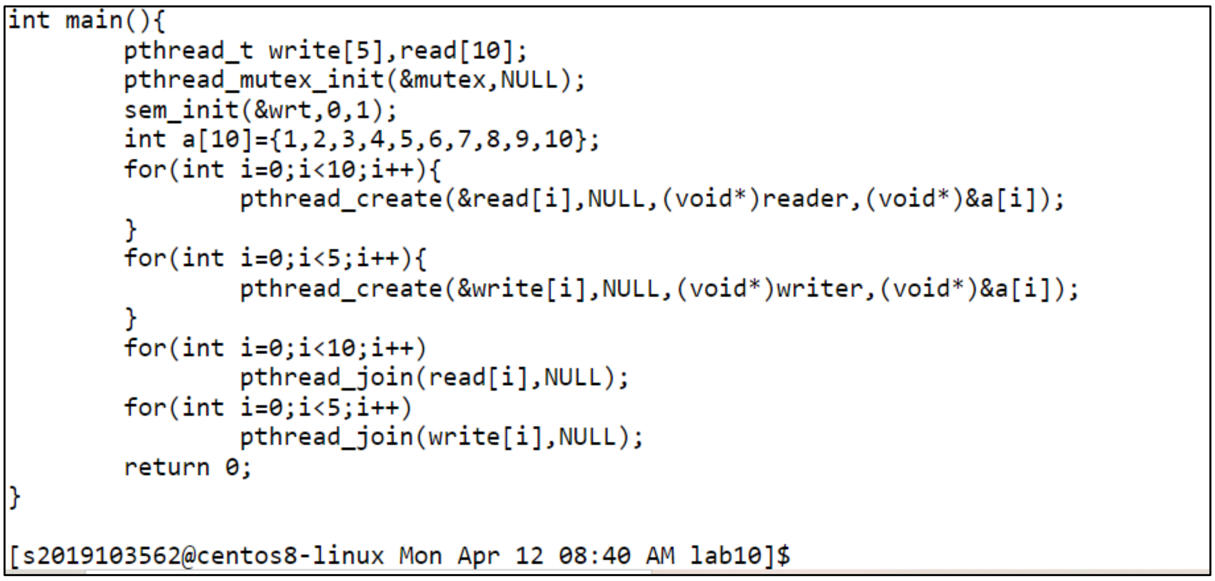
READER'S WRITER'S PROBLEM USING SEMAPHORE SYSTEM CALLS



```
[s2019103562@centos8-linux Mon Apr 12 08:37 AM lab10]$ vim rw.c
[s2019103562@centos8-linux Mon Apr 12 08:37 AM lab10]$ gcc rw.c -lpthread -lrt
[s2019103562@centos8-linux Mon Apr 12 08:37 AM lab10]$ ./a.out
Reader 1 read var as 1
Reader 2 read var as 1
Reader 4 read var as 1
Reader 6 read var as 1
Reader 7 read var as 1
Reader 5 read var as 1
Reader 8 read var as 1
Reader 3 read var as 1
Reader 9 read var as 1
Reader 10 read var as 1
Writer 1 modified var to 2
Writer 2 modified var to 4
Writer 3 modified var to 8
Writer 4 modified var to 16
Writer 5 modified var to 32
[s2019103562@centos8-linux Mon Apr 12 08:38 AM lab10]$ 
```

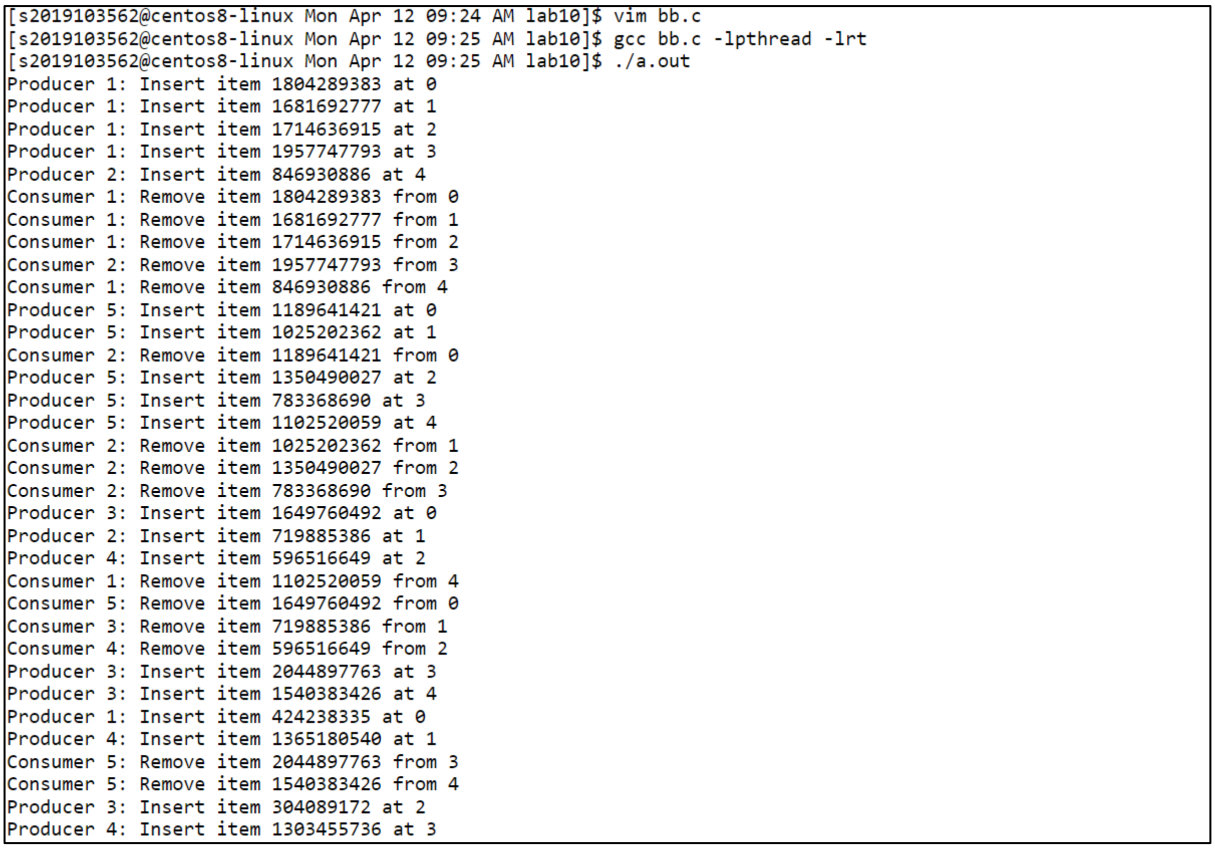


```
[s2019103562@centos8-linux Mon Apr 12 08:38 AM lab10]$ cat rw.c
#include<pthread.h>
#include<semaphore.h>
#include<stdio.h>
sem_t wrt;
pthread_mutex_t mutex;
int var=1;
int numreader=0;
void* writer(void* wno){
        sem_wait(&wrt);
        var=var*2;
        printf("Writer %d modified var to %d\n",(*(int*)wno),var);
        sem_post(&wrt);
}
void* reader(void* rno){
        pthread_mutex_lock(&mutex);
        numreader++;
        if(numreader==1){
                sem_wait(&wrt);
        }
        pthread_mutex_unlock(&mutex);
        printf("Reader %d read var as %d\n",(*(int*)rno),var);
        pthread_mutex_lock(&mutex);
        numreader--;
        if(numreader==0){
                sem_post(&wrt);
        }
        pthread_mutex_unlock(&mutex);
}
```

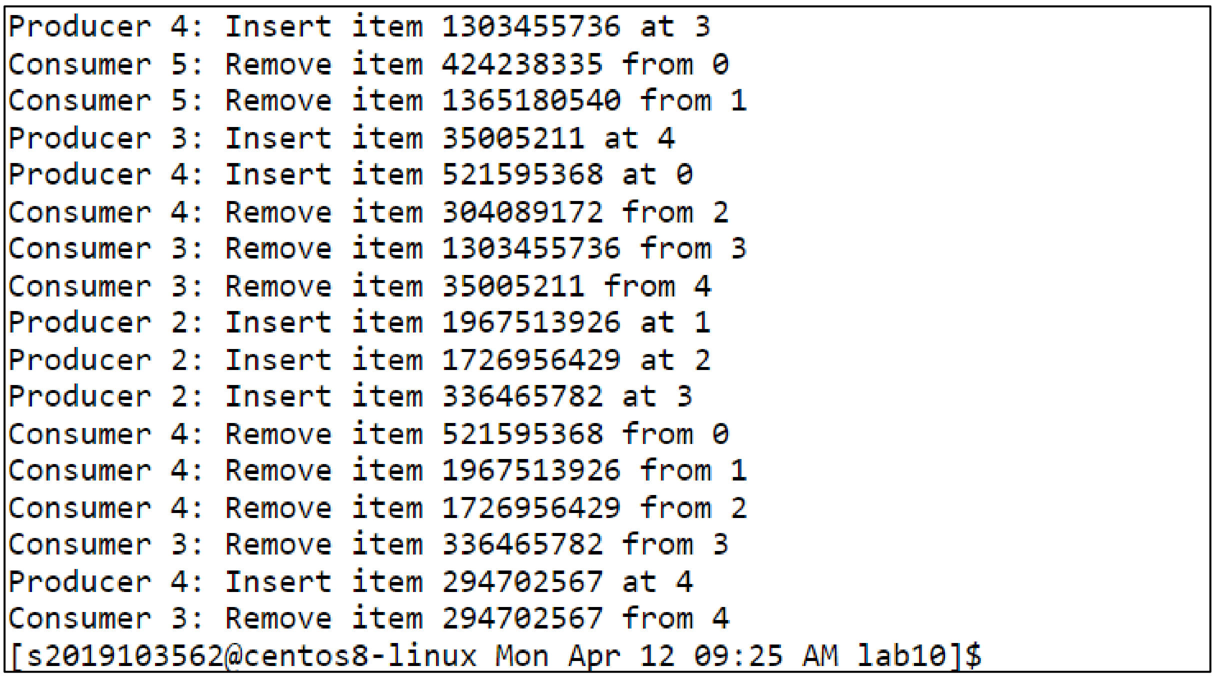


```
int main(){
        pthread_t write[5],read[10];
        pthread_mutex_init(&mutex,NULL);
        sem_init(&wrt,0,1);
        int a[10]={1,2,3,4,5,6,7,8,9,10};
        for(int i=0;i<10;i++){
                pthread_create(&read[i],NULL,(void*)reader,(void*)&a[i]);
        }
        for(int i=0;i<5;i++){
                pthread_create(&write[i],NULL,(void*)writer,(void*)&a[i]);
        }
        for(int i=0;i<10;i++)
                pthread_join(read[i],NULL);
        for(int i=0;i<5;i++)
                pthread_join(write[i],NULL);
        return 0;
}

[s2019103562@centos8-linux Mon Apr 12 08:40 AM lab10]$
```

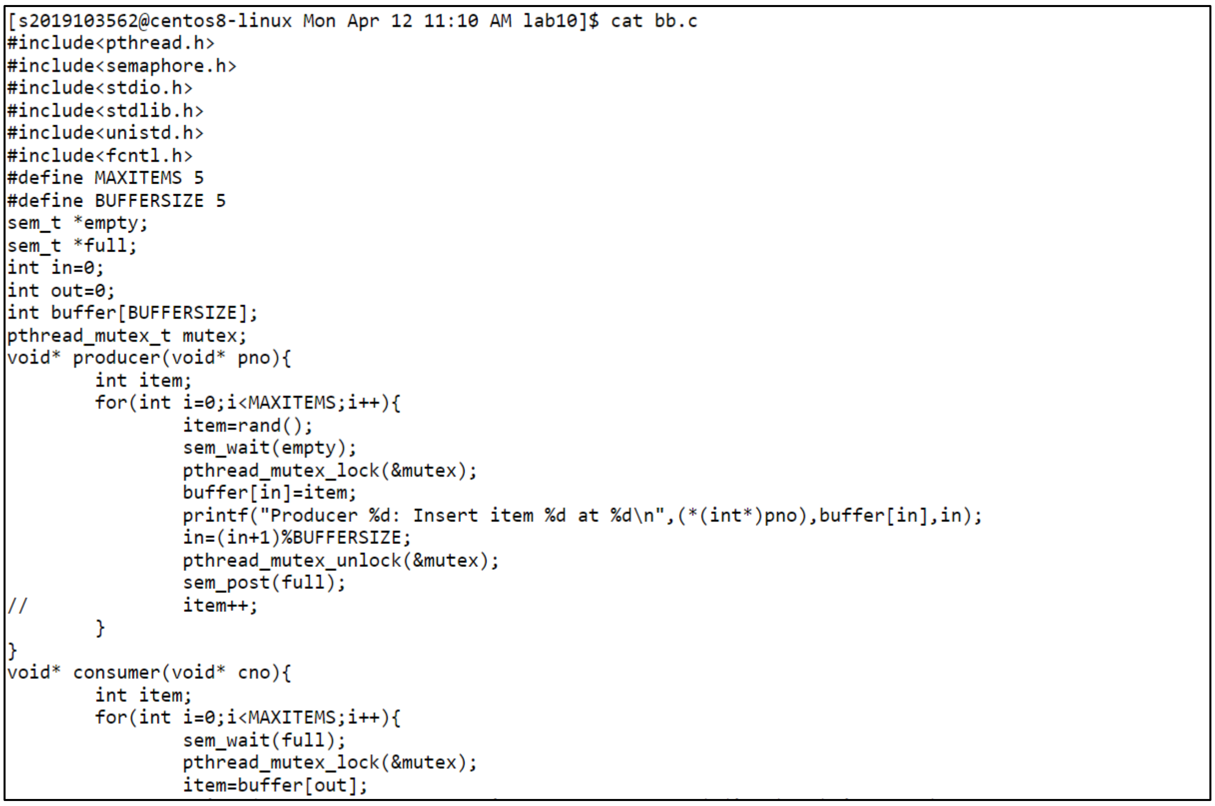
PRODUCER CONSUMER PROBLEM USING SEMAPHORE SYSTEM CALLS(BOUNDED BUFFER):



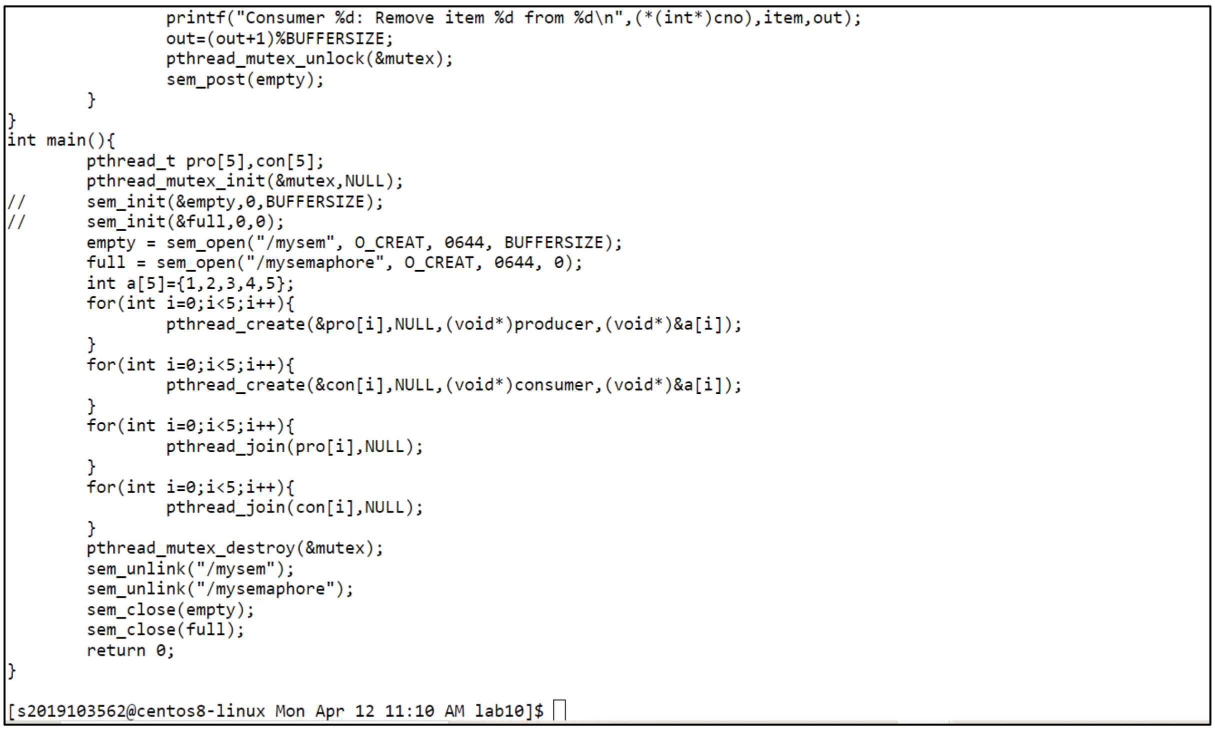
```
[s2019103562@centos8-linux Mon Apr 12 09:24 AM lab10]$ vim bb.c
[s2019103562@centos8-linux Mon Apr 12 09:25 AM lab10]$ gcc bb.c -lpthread -lrt
[s2019103562@centos8-linux Mon Apr 12 09:25 AM lab10]$ ./a.out
Producer 1: Insert item 1804289383 at 0
Producer 1: Insert item 1681692777 at 1
Producer 1: Insert item 1714636915 at 2
Producer 1: Insert item 1957747793 at 3
Producer 2: Insert item 846930886 at 4
Consumer 1: Remove item 1804289383 from 0
Consumer 1: Remove item 1681692777 from 1
Consumer 1: Remove item 1714636915 from 2
Consumer 2: Remove item 1957747793 from 3
Consumer 1: Remove item 846930886 from 4
Producer 5: Insert item 1189641421 at 0
Producer 5: Insert item 1025202362 at 1
Consumer 2: Remove item 1189641421 from 0
Producer 5: Insert item 1350490027 at 2
Producer 5: Insert item 783368690 at 3
Producer 5: Insert item 1102520059 at 4
Consumer 2: Remove item 1025202362 from 1
Consumer 2: Remove item 1350490027 from 2
Consumer 2: Remove item 783368690 from 3
Producer 3: Insert item 1649760492 at 0
Producer 2: Insert item 719885386 at 1
Producer 4: Insert item 596516649 at 2
Consumer 1: Remove item 1102520059 from 4
Consumer 5: Remove item 1649760492 from 0
Consumer 3: Remove item 719885386 from 1
Consumer 4: Remove item 596516649 from 2
Producer 3: Insert item 2044897763 at 3
Producer 3: Insert item 1540383426 at 4
Producer 1: Insert item 424238335 at 0
Producer 4: Insert item 1365180540 at 1
Consumer 5: Remove item 2044897763 from 3
Consumer 5: Remove item 1540383426 from 4
Producer 3: Insert item 304089172 at 2
Producer 4: Insert item 1303455736 at 3
```



```
Producer 4: Insert item 1303455736 at 3
Consumer 5: Remove item 424238335 from 0
Consumer 5: Remove item 1365180540 from 1
Producer 3: Insert item 35005211 at 4
Producer 4: Insert item 521595368 at 0
Consumer 4: Remove item 304089172 from 2
Consumer 3: Remove item 1303455736 from 3
Consumer 3: Remove item 35005211 from 4
Producer 2: Insert item 1967513926 at 1
Producer 2: Insert item 1726956429 at 2
Producer 2: Insert item 336465782 at 3
Consumer 4: Remove item 521595368 from 0
Consumer 4: Remove item 1967513926 from 1
Consumer 4: Remove item 1726956429 from 2
Consumer 3: Remove item 336465782 from 3
Producer 4: Insert item 294702567 at 4
Consumer 3: Remove item 294702567 from 4
[s2019103562@centos8-linux Mon Apr 12 09:25 AM lab10]$
```

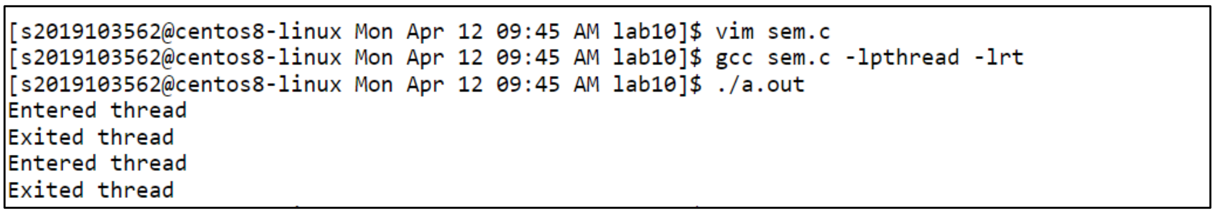


```
[s2019103562@centos8-linux Mon Apr 12 11:10 AM lab10]$ cat bb.c
#include<pthread.h>
#include<semaphore.h>
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<fcntl.h>
#define MAXITEMS 5
#define BUFFERSIZE 5
sem_t *empty;
sem_t *full;
int in=0;
int out=0;
int buffer[BUFFERSIZE];
pthread_mutex_t mutex;
void* producer(void* pno){
        int item;
        for(int i=0;i<MAXITEMS;i++){
                item=rand();
                sem_wait(empty);
                pthread_mutex_lock(&mutex);
                buffer[in]=item;
                printf("Producer %d: Insert item %d at %d\n",(*(int*)pno),buffer[in],in);
                in=(in+1)%BUFFERSIZE;
                pthread_mutex_unlock(&mutex);
                sem_post(full);
//              item++;
        }
}
void* consumer(void* cno){
        int item;
        for(int i=0;i<MAXITEMS;i++){
                sem_wait(full);
                pthread_mutex_lock(&mutex);
                item=buffer[out];
```

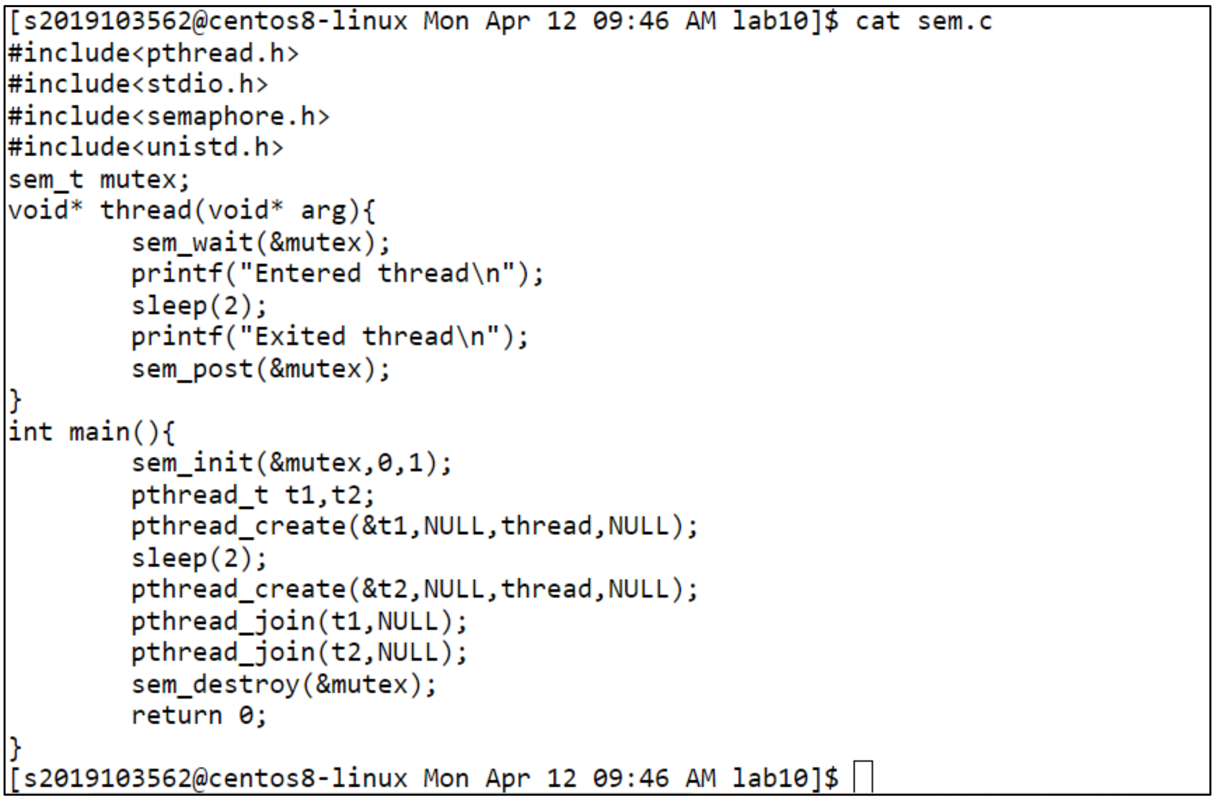


```
                printf("Consumer %d: Remove item %d from %d\n",(*(int*)cno),item,out);
                out=(out+1)%BUFFERSIZE;
                pthread_mutex_unlock(&mutex);
                sem_post(empty);
        }
}
int main(){
        pthread_t pro[5],con[5];
        pthread_mutex_init(&mutex,NULL);
//      sem_init(&empty,0,BUFFERSIZE);
//      sem_init(&full,0,0);
        empty = sem_open("/mysem", O_CREAT, 0644, BUFFERSIZE);
        full = sem_open("/mysemaphore", O_CREAT, 0644, 0);
        int a[5]={1,2,3,4,5};
        for(int i=0;i<5;i++){
                pthread_create(&pro[i],NULL,(void*)producer,(void*)&a[i]);
        }
        for(int i=0;i<5;i++){
                pthread_create(&con[i],NULL,(void*)consumer,(void*)&a[i]);
        }
        for(int i=0;i<5;i++){
                pthread_join(pro[i],NULL);
        }
        for(int i=0;i<5;i++){
                pthread_join(con[i],NULL);
        }
        pthread_mutex_destroy(&mutex);
        sem_unlink("/mysem");
        sem_unlink("/mysemaphore");
        sem_close(empty);
        sem_close(full);
        return 0;
}

[s2019103562@centos8-linux Mon Apr 12 11:10 AM lab10]$ 
```

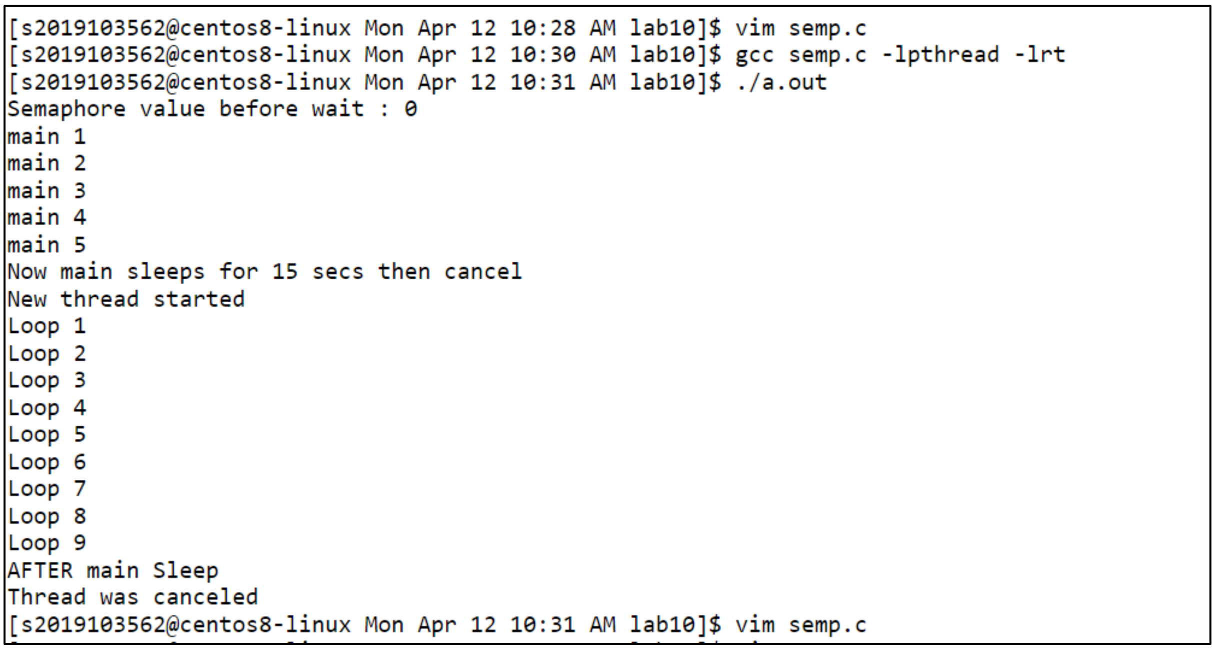
SEM WAIT



```
[s2019103562@centos8-linux Mon Apr 12 09:45 AM lab10]$ vim sem.c
[s2019103562@centos8-linux Mon Apr 12 09:45 AM lab10]$ gcc sem.c -lpthread -lrt
[s2019103562@centos8-linux Mon Apr 12 09:45 AM lab10]$ ./a.out
Entered thread
Exited thread
Entered thread
Exited thread
```

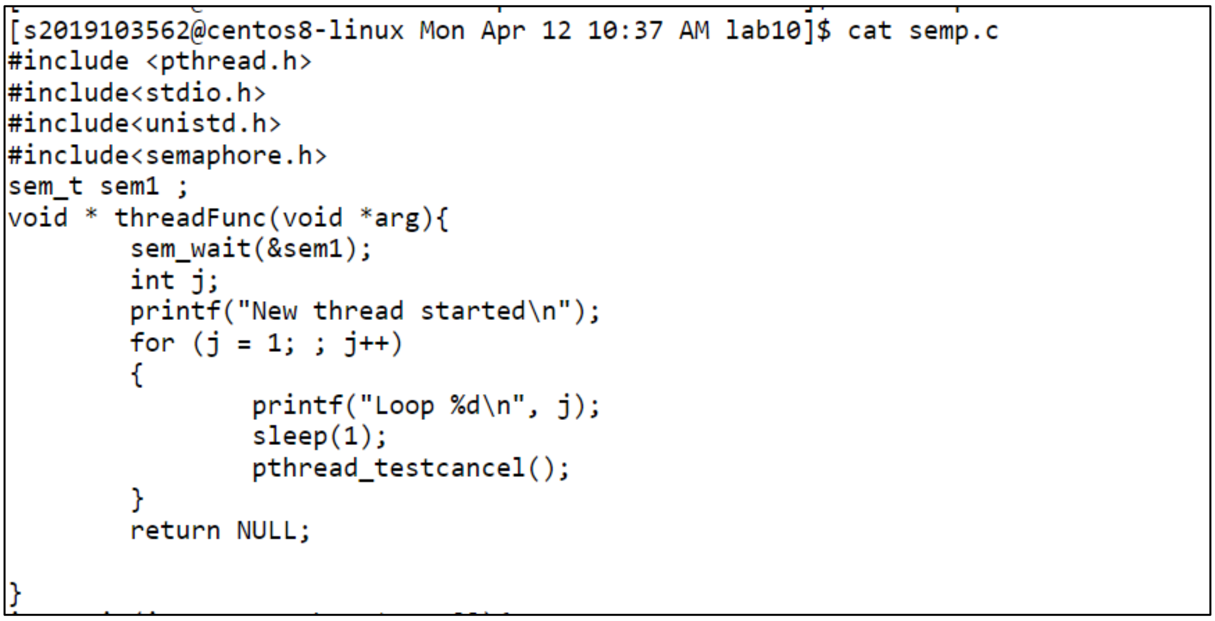


```
[s2019103562@centos8-linux Mon Apr 12 09:46 AM lab10]$ cat sem.c
#include<pthread.h>
#include<stdio.h>
#include<semaphore.h>
#include<unistd.h>
sem_t mutex;
void* thread(void* arg){
        sem_wait(&mutex);
        printf("Entered thread\n");
        sleep(2);
        printf("Exited thread\n");
        sem_post(&mutex);
}
int main(){
        sem_init(&mutex,0,1);
        pthread_t t1,t2;
        pthread_create(&t1,NULL,thread,NULL);
        sleep(2);
        pthread_create(&t2,NULL,thread,NULL);
        pthread_join(t1,NULL);
        pthread_join(t2,NULL);
        sem_destroy(&mutex);
        return 0;
}
[s2019103562@centos8-linux Mon Apr 12 09:46 AM lab10]$
```

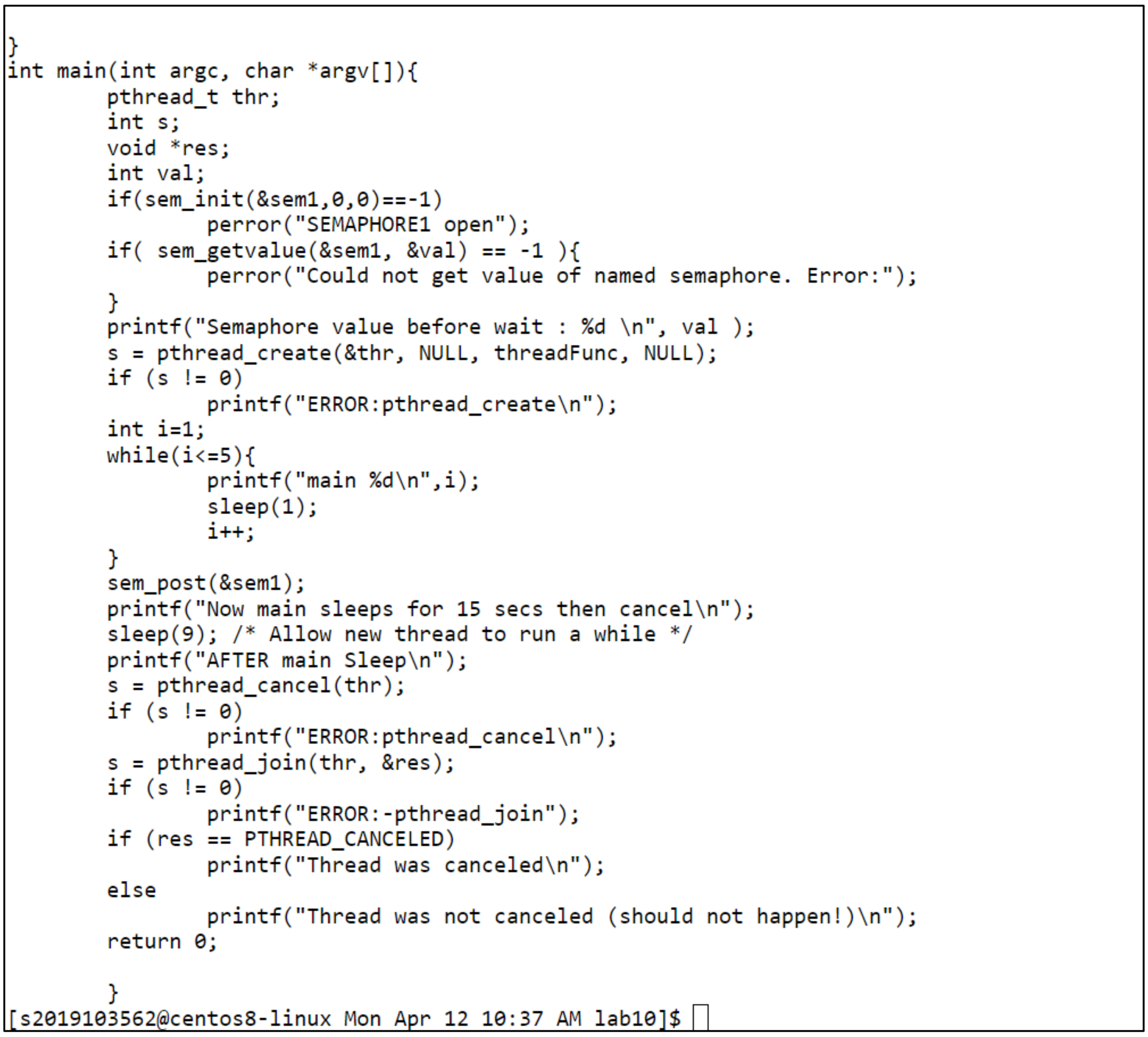


```
[s2019103562@centos8-linux Mon Apr 12 10:28 AM lab10]$ vim semp.c
[s2019103562@centos8-linux Mon Apr 12 10:30 AM lab10]$ gcc semp.c -lpthread -lrt
[s2019103562@centos8-linux Mon Apr 12 10:31 AM lab10]$ ./a.out
Semaphore value before wait : 0
main 1
main 2
main 3
main 4
main 5
Now main sleeps for 15 secs then cancel
New thread started
Loop 1
Loop 2
Loop 3
Loop 4
Loop 5
Loop 6
Loop 7
Loop 8
Loop 9
AFTER main Sleep
Thread was canceled
[s2019103562@centos8-linux Mon Apr 12 10:31 AM lab10]$ vim semp.c
```

PTHREAD_CANCEL

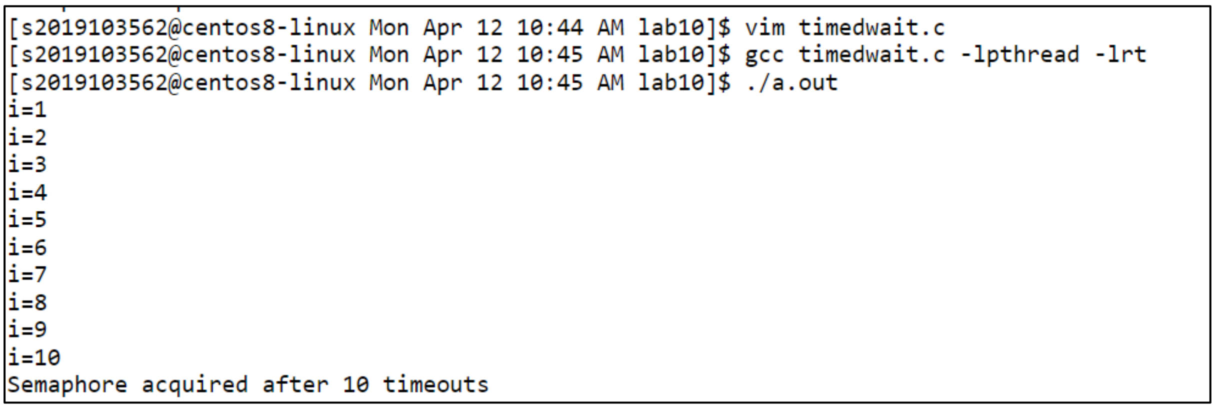


```
[s2019103562@centos8-linux Mon Apr 12 10:37 AM lab10]$ cat semp.c
#include <pthread.h>
#include<stdio.h>
#include<unistd.h>
#include<semaphore.h>
sem_t sem1 ;
void * threadFunc(void *arg){
        sem_wait(&sem1);
        int j;
        printf("New thread started\n");
        for (j = 1; ; j++)
        {
                printf("Loop %d\n", j);
                sleep(1);
                pthread_testcancel();
        }
        return NULL;

}
```

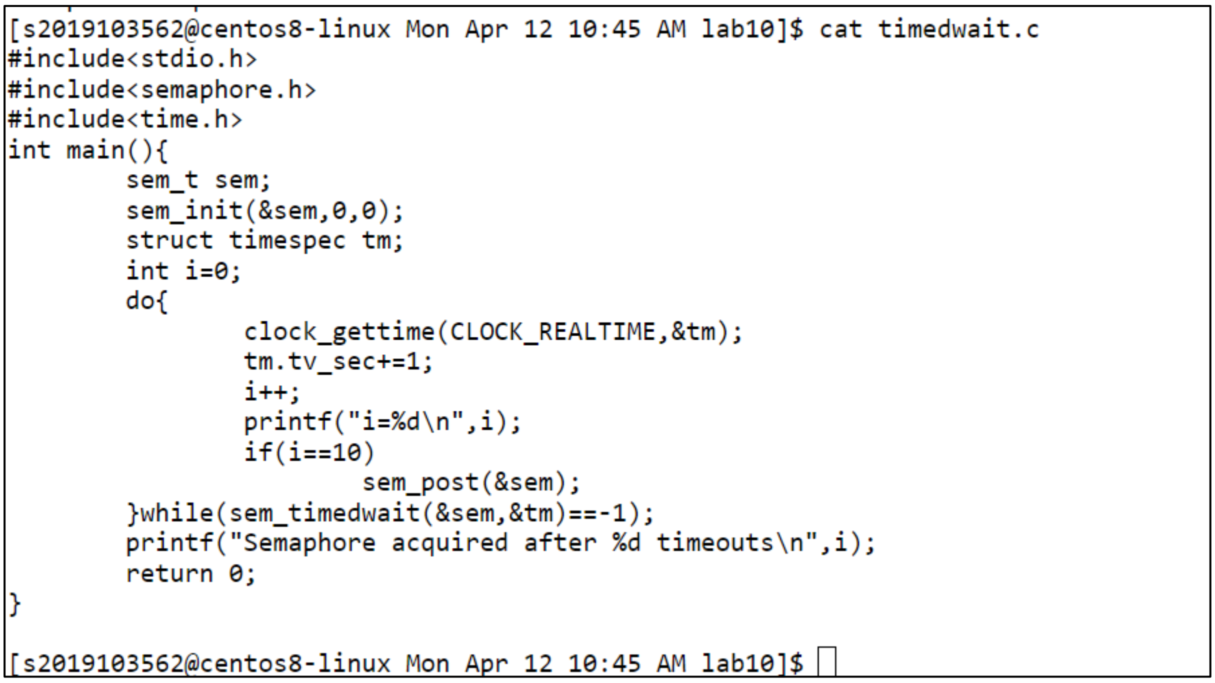


```

}
int main(int argc, char *argv[]){
        pthread_t thr;
        int s;
        void *res;
        int val;
        if(sem_init(&sem1,0,0)==-1)
                perror("SEMAPHORE1 open");
        if( sem_getvalue(&sem1, &val) == -1 ){
                perror("Could not get value of named semaphore. Error:");
        }
        printf("Semaphore value before wait : %d \n", val );
        s = pthread_create(&thr, NULL, threadFunc, NULL);
        if (s != 0)
                printf("ERROR:pthread_create\n");
        int i=1;
        while(i<=5){
                printf("main %d\n",i);
                sleep(1);
                i++;
        }
        sem_post(&sem1);
        printf("Now main sleeps for 15 secs then cancel\n");
        sleep(9); /* Allow new thread to run a while */
        printf("AFTER main Sleep\n");
        s = pthread_cancel(thr);
        if (s != 0)
                printf("ERROR:pthread_cancel\n");
        s = pthread_join(thr, &res);
        if (s != 0)
                printf("ERROR:-pthread_join");
        if (res == PTHREAD_CANCELED)
                printf("Thread was canceled\n");
        else
                printf("Thread was not canceled (should not happen!)\n");
        return 0;

        }
[s2019103562@centos8-linux Mon Apr 12 10:37 AM lab10]$
```

TIMED WAIT

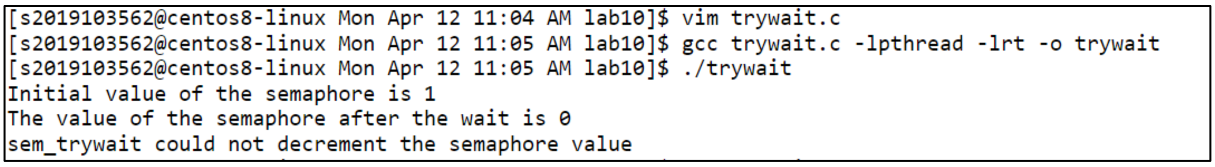


```
[s2019103562@centos8-linux Mon Apr 12 10:44 AM lab10]$ vim timedwait.c
[s2019103562@centos8-linux Mon Apr 12 10:45 AM lab10]$ gcc timedwait.c -lpthread -lrt
[s2019103562@centos8-linux Mon Apr 12 10:45 AM lab10]$ ./a.out
i=1
i=2
i=3
i=4
i=5
i=6
i=7
i=8
i=9
i=10
Semaphore acquired after 10 timeouts
```

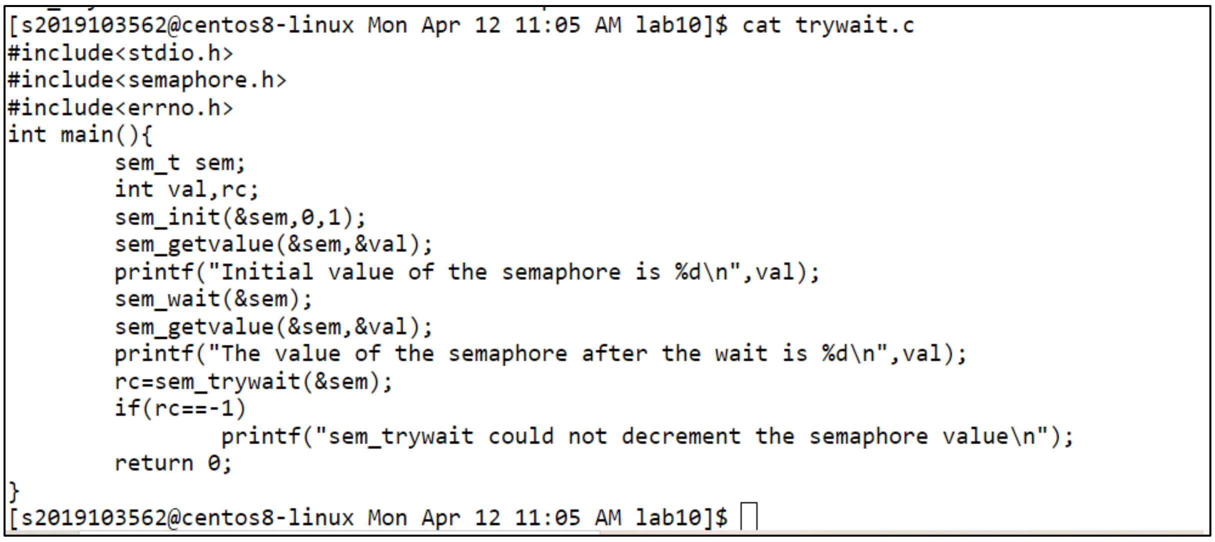


```
[s2019103562@centos8-linux Mon Apr 12 10:45 AM lab10]$ cat timedwait.c
#include<stdio.h>
#include<semaphore.h>
#include<time.h>
int main(){
        sem_t sem;
        sem_init(&sem,0,0);
        struct timespec tm;
        int i=0;
        do{
                clock_gettime(CLOCK_REALTIME,&tm);
                tm.tv_sec+=1;
                i++;
                printf("i=%d\n",i);
                if(i==10)
                        sem_post(&sem);
        }while(sem_timedwait(&sem,&tm)==-1);
        printf("Semaphore acquired after %d timeouts\n",i);
        return 0;
}

[s2019103562@centos8-linux Mon Apr 12 10:45 AM lab10]$
```

TRY WAIT

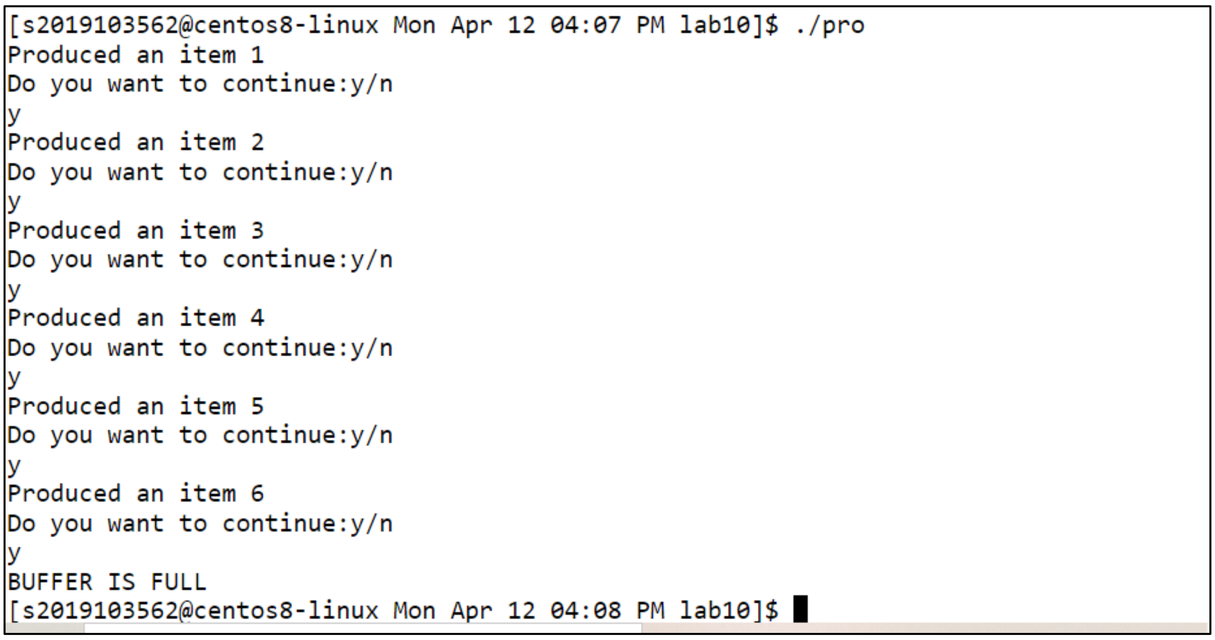


```
[s2019103562@centos8-linux Mon Apr 12 11:04 AM lab10]$ vim trywait.c
[s2019103562@centos8-linux Mon Apr 12 11:05 AM lab10]$ gcc trywait.c -lpthread -lrt -o trywait
[s2019103562@centos8-linux Mon Apr 12 11:05 AM lab10]$ ./trywait
Initial value of the semaphore is 1
The value of the semaphore after the wait is 0
sem_trywait could not decrement the semaphore value
```

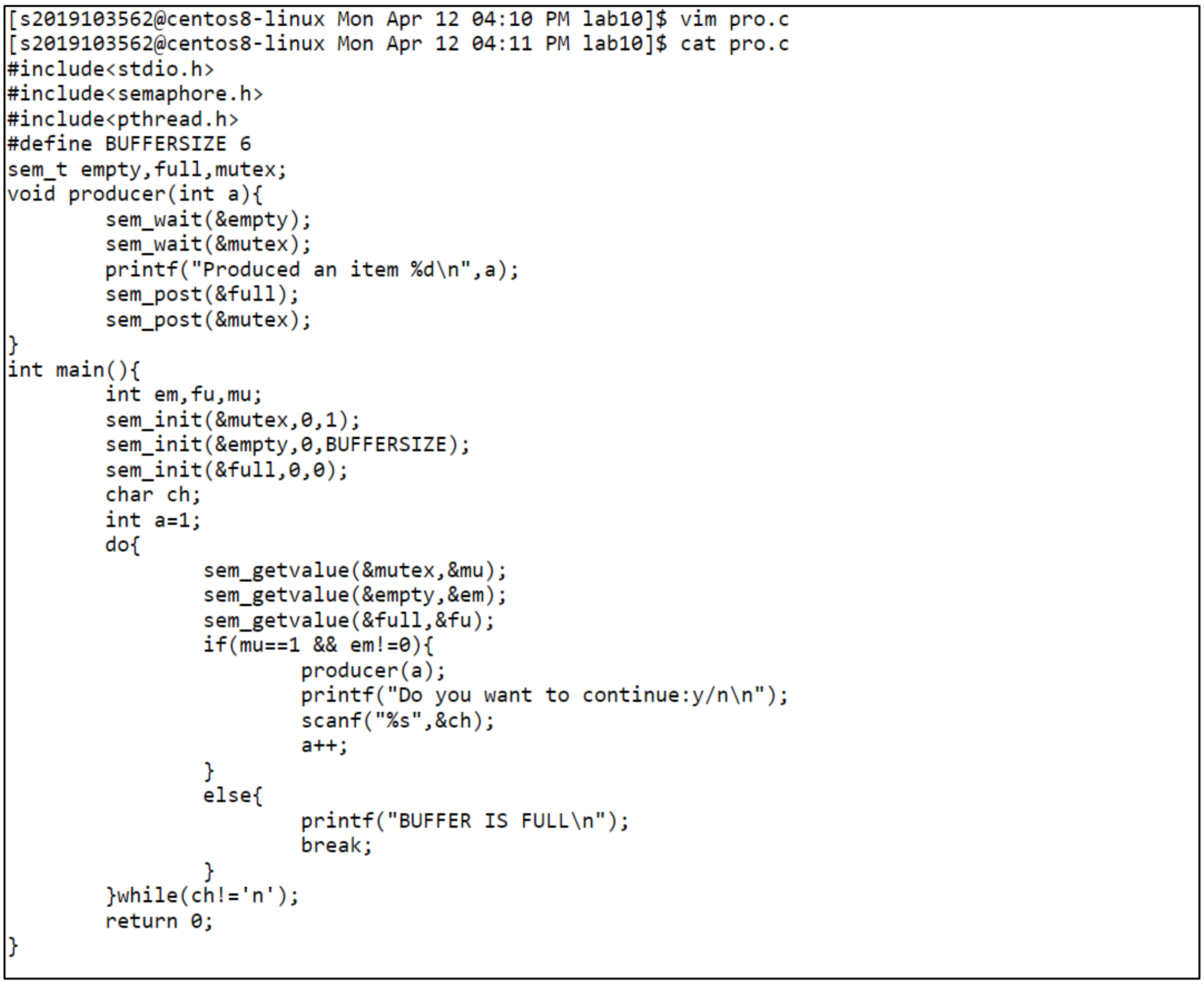


```
[s2019103562@centos8-linux Mon Apr 12 11:05 AM lab10]$ cat trywait.c
#include<stdio.h>
#include<semaphore.h>
#include<errno.h>
int main(){
        sem_t sem;
        int val,rc;
        sem_init(&sem,0,1);
        sem_getvalue(&sem,&val);
        printf("Initial value of the semaphore is %d\n",val);
        sem_wait(&sem);
        sem_getvalue(&sem,&val);
        printf("The value of the semaphore after the wait is %d\n",val);
        rc=sem_trywait(&sem);
        if(rc==-1)
                printf("sem_trywait could not decrement the semaphore value\n");
        return 0;
}
[s2019103562@centos8-linux Mon Apr 12 11:05 AM lab10]$
```

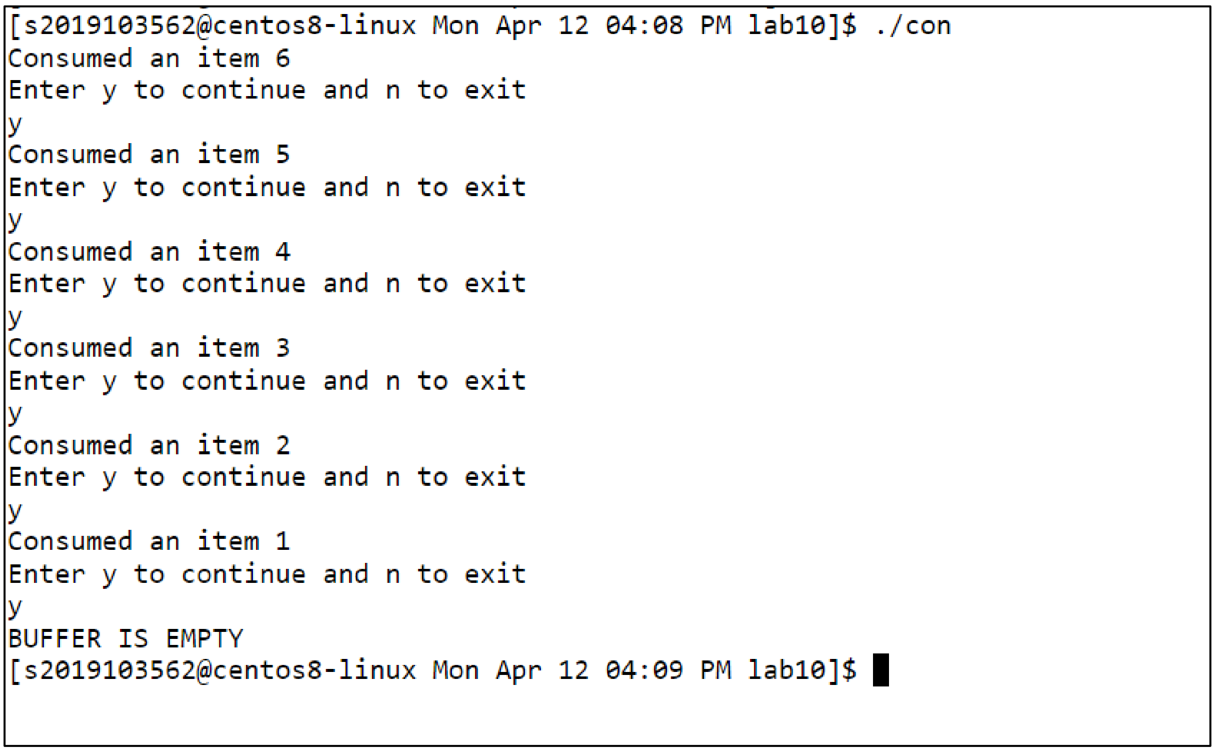
PRODUCER CONSUMER PROBLEM(SEPARATE PROGRAM FOR PRODUCER AND CONSUMER)



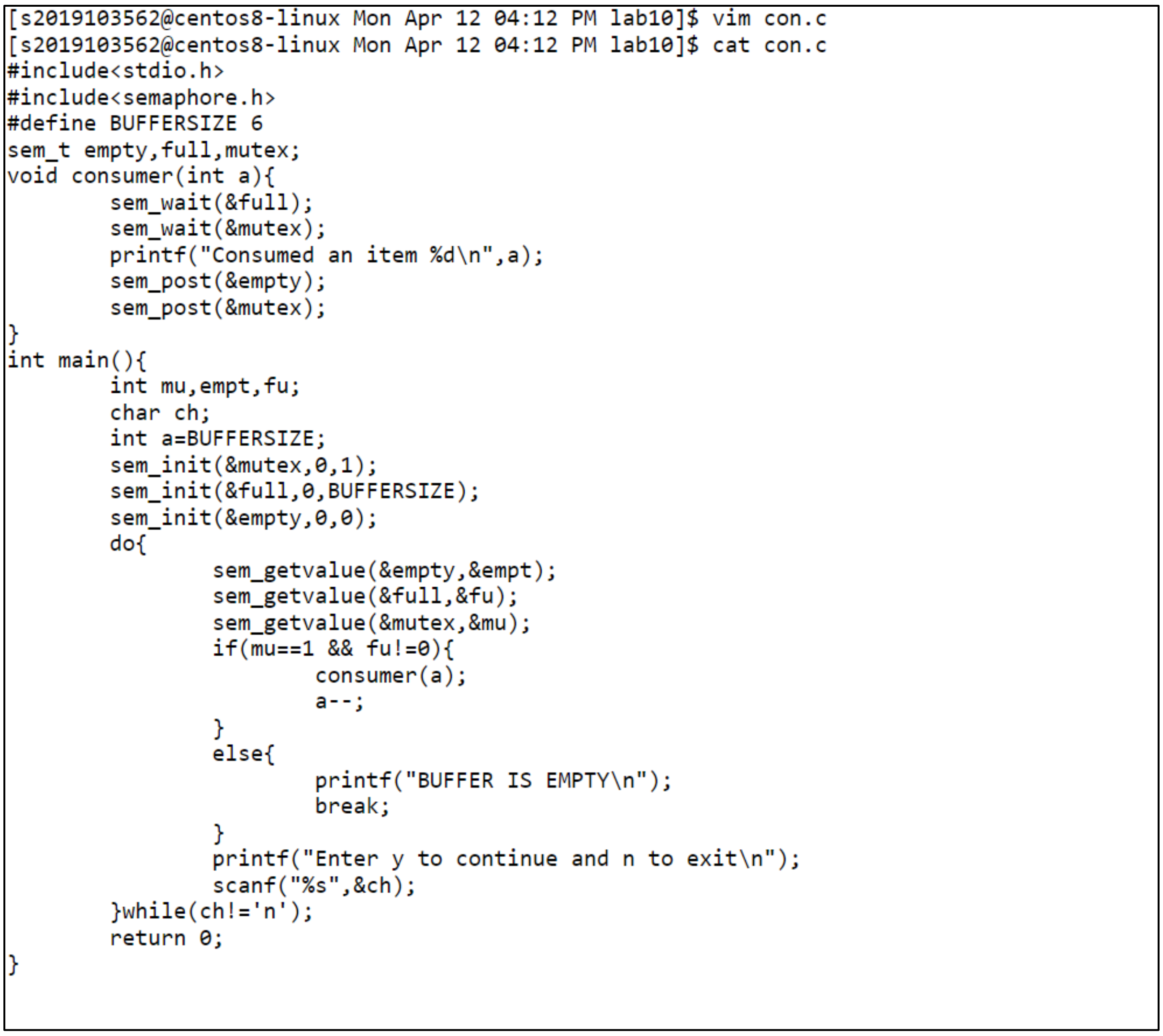
```
[s2019103562@centos8-linux Mon Apr 12 04:07 PM lab10]$ ./pro
Produced an item 1
Do you want to continue:y/n
y
Produced an item 2
Do you want to continue:y/n
y
Produced an item 3
Do you want to continue:y/n
y
Produced an item 4
Do you want to continue:y/n
y
Produced an item 5
Do you want to continue:y/n
y
Produced an item 6
Do you want to continue:y/n
y
BUFFER IS FULL
[s2019103562@centos8-linux Mon Apr 12 04:08 PM lab10]$
```



```
[s2019103562@centos8-linux Mon Apr 12 04:10 PM lab10]$ vim pro.c
[s2019103562@centos8-linux Mon Apr 12 04:11 PM lab10]$ cat pro.c
#include<stdio.h>
#include<semaphore.h>
#include<pthread.h>
#define BUFFERSIZE 6
sem_t empty,full,mutex;
void producer(int a){
        sem_wait(&empty);
        sem_wait(&mutex);
        printf("Produced an item %d\n",a);
        sem_post(&full);
        sem_post(&mutex);
}
int main(){
        int em,fu,mu;
        sem_init(&mutex,0,1);
        sem_init(&empty,0,BUFFERSIZE);
        sem_init(&full,0,0);
        char ch;
        int a=1;
        do{
                sem_getvalue(&mutex,&mu);
                sem_getvalue(&empty,&em);
                sem_getvalue(&full,&fu);
                if(mu==1 && em!=0){
                        producer(a);
                        printf("Do you want to continue:y/n\n");
                        scanf("%s",&ch);
                        a++;
                }
                else{
                        printf("BUFFER IS FULL\n");
                        break;
                }
        }while(ch!='n');
        return 0;
}
```



```
[s2019103562@centos8-linux Mon Apr 12 04:08 PM lab10]$ ./con
Consumed an item 6
Enter y to continue and n to exit
y
Consumed an item 5
Enter y to continue and n to exit
y
Consumed an item 4
Enter y to continue and n to exit
y
Consumed an item 3
Enter y to continue and n to exit
y
Consumed an item 2
Enter y to continue and n to exit
y
Consumed an item 1
Enter y to continue and n to exit
y
BUFFER IS EMPTY
[s2019103562@centos8-linux Mon Apr 12 04:09 PM lab10]$ █
```



```
[s2019103562@centos8-linux Mon Apr 12 04:12 PM lab10]$ vim con.c
[s2019103562@centos8-linux Mon Apr 12 04:12 PM lab10]$ cat con.c
#include<stdio.h>
#include<semaphore.h>
#define BUFFERSIZE 6
sem_t empty,full,mutex;
void consumer(int a){
        sem_wait(&full);
        sem_wait(&mutex);
        printf("Consumed an item %d\n",a);
        sem_post(&empty);
        sem_post(&mutex);
}
int main(){
        int mu,empt,fu;
        char ch;
        int a=BUFFERSIZE;
        sem_init(&mutex,0,1);
        sem_init(&full,0,BUFFERSIZE);
        sem_init(&empty,0,0);
        do{
                sem_getvalue(&empty,&empt);
                sem_getvalue(&full,&fu);
                sem_getvalue(&mutex,&mu);
                if(mu==1 && fu!=0){
                        consumer(a);
                        a--;
                }
                else{
                        printf("BUFFER IS EMPTY\n");
                        break;
                }
                printf("Enter y to continue and n to exit\n");
                scanf("%s",&ch);
        }while(ch!='n');
        return 0;
}
```