OPERATING SYSTEMS

SEMAPHORES

COE18B043

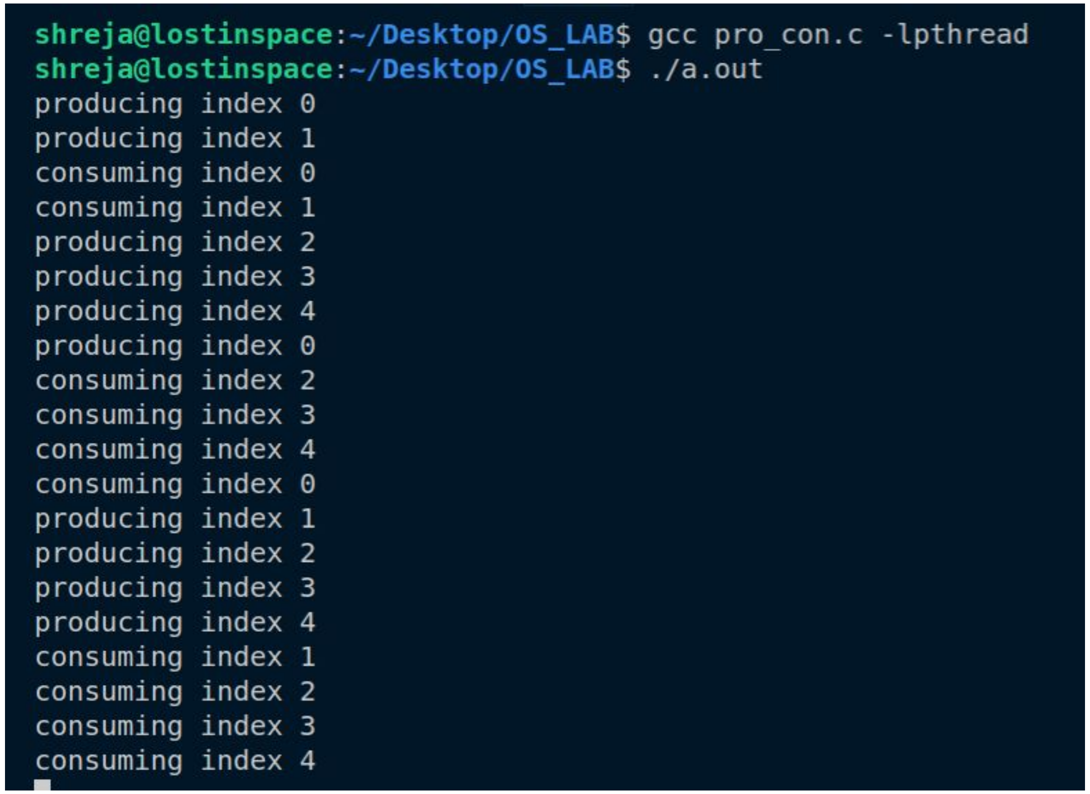
ASSIGNMENT 7

R SHREJA

1.Simulate the Producer-Consumer code discussed in the class

```
#include<stdio.h>
#include<sys/wait.h>
#include<pthread.h>
#define N 5

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

int buf[N];
int in=0,out=0;
int main()
{
   pthread t tid[2];

   pthread create(&tid[0],NULL,producer,NULL);

   pthread create(&tid[1],NULL,consumer,NULL);

   pthread join(tid[0],NULL);

   pthread join(tid[1],NULL);

   return 0;

}


void *producer(void *param)

{

   int data=3,i=0;

   while(i<10)

   {

       while((in+1)%N==out);


       buf[in]=data;

       printf("producing index %d\n",in);

       in=(in+1)%N;

       ++i;

   }
   pthread exit(0);
}


void *consumer(void *param)
{
   int data;
   while(1)
   {
       while(in==out);

       data=buf[out];
       printf("consuming index %d\n",out);
       out=(out+1)%N;
   }
   pthread exit(0);
}
```

OUTPUT:



```
shreja@lostinspace:~/Desktop/OS_LAB$ gcc pro_con.c -lpthread
shreja@lostinspace:~/Desktop/OS_LAB$ ./a.out
producing index 0
producing index 1
consuming index 0
consuming index 1
producing index 2
producing index 3
producing index 4
producing index 0
consuming index 2
consuming index 3
consuming index 4
consuming index 0
producing index 1
producing index 2
producing index 3
producing index 4
consuming index 1
consuming index 2
consuming index 3
consuming index 4
```

2. Extend the producer-consumer simulation in Q1 to sync access of critical data using Peterson's algorithm.

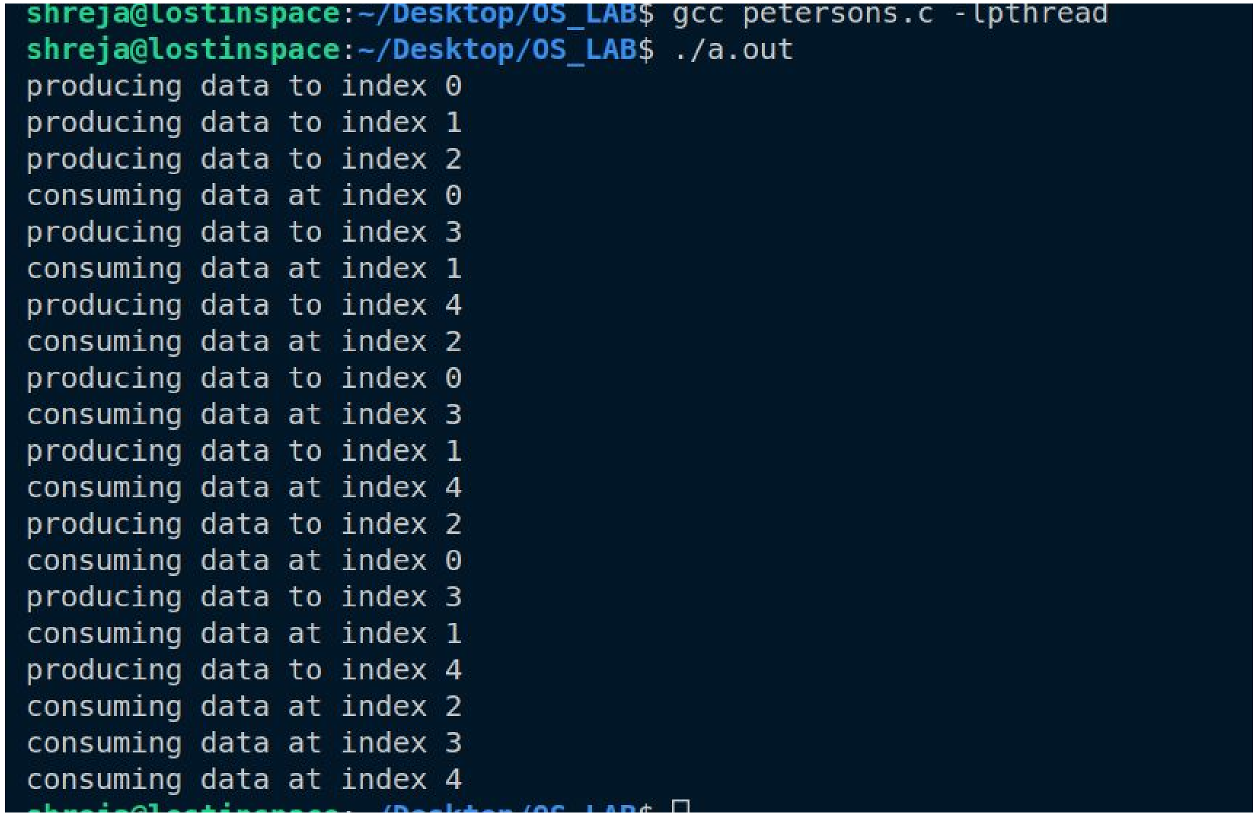
```
#include <stdio.h>
#include <sys/wait.h>
#include <pthread.h>
#define N 5
void *producer(void *args);
void *consumer(void *args);
int buf[N];
int in = 0, out = 0,counter=0;
int flag[] = {0, 0};
int turn = 0;
// acquire lock
void lock(int id)
{
   // set the flag to let the other thread know that this thread wants to acquire a lock
   flag[id] = 1;
```

```
    // give the other thread a chance first
    turn = 1 - id;
    // wait till the other thread finishes
    while (flag[1 - id] == 1 && turn == 1 - id)
        ;
}
// release lock
void unlock(int id)
{
    // set the flag to let the other thread know that this thread does not need the lock anymore
    flag[id] = 0;
}

int main(){
    pthread t tid[2];
    pthread create(&tid[0], NULL, producer, NULL);
    pthread create(&tid[1], NULL, consumer, NULL);
    pthread join(tid[0], NULL);
    pthread join(tid[1], NULL);
    return 0;
```

```
}


void *producer(void *args){
   int data = 3, i = 0;
   while (i < 10){
       lock(0);
       buf[in] = data;
       printf("producing data to index %d\n", in);
       in = (in + 1) % N;
       i++;
       counter++;
       unlock(0);
   }
   pthread exit(0);
}
void *consumer(void *args){
   int data, i = 0;
   while (i < 10){
       while(counter==0);
       lock(1);
```

```
        data = buf[out];

        printf("consuming data at index %d\n", out);

        out = (out + 1) % N;

        i++;

        counter--;

        unlock(1);

    }

    pthread exit(0);

}
```

OUTPUT:



```
shreja@lostinspace:~/Desktop/OS_LAB$ gcc petersons.c -lpthread
shreja@lostinspace:~/Desktop/OS_LAB$ ./a.out
producing data to index 0
producing data to index 1
producing data to index 2
consuming data at index 0
producing data to index 3
consuming data at index 1
producing data to index 4
consuming data at index 2
producing data to index 0
consuming data at index 3
producing data to index 1
consuming data at index 4
producing data to index 2
consuming data at index 0
producing data to index 3
consuming data at index 1
producing data to index 4
consuming data at index 2
consuming data at index 3
consuming data at index 4
shreja@lostinspace:~/Desktop/OS_LAB$
```

3. Dictionary Problem: Let the producer set up a dictionary of at least 20 words with three attributes(Word, Primary meaning, Secondary meaning) and let the consumer search for the word and retrieve its respective primary and secondary meaning.

```c
#include <stdio.h>

#include <sys/wait.h>

#include <pthread.h>

#include <string.h>

#define N 20

void *producer(void *args);

void *consumer(void *args);

struct dict

{

   char word[30];

   char meaning_1[100], meaning_2[100];

} buf[N];


int flag[] = {0, 0};

int turn = 0;

char search_word[30];
```

```
int in = 0, out = 0, end = 0;
int j = 0;

// acquire lock
void lock(int id)
{
   // set the flag to let the other thread know that this thread wants to acquire a lock
   flag[id] = 1;
   // give the other thread a chance first
   turn = 1 - id;
   // wait till the other thread finishes
   while (flag[1 - id] == 1 && turn == 1 - id)
       ;
}
// release lock
void unlock(int id)
{
   // set the flag to let the other thread know that this thread does not need the lock anymore
   flag[id] = 0;
}
```

```
int main(int argc, char *argv[])
{
   pthread t tid[2];
   strcpy(search word, argv[1]);
   pthread create(&tid[0], NULL, producer, NULL);
   pthread create(&tid[1], NULL, consumer, NULL);
   pthread join(tid[0], NULL);
   pthread join(tid[1], NULL);
   return 0;
}
void *producer(void *args)
{
   int i = 0;
   FILE *fptr = fopen("dict.txt", "r");
   while (1)
   {
       lock(0);
       j = 0;
       while (j < N)
       {
```

```
char ch;
char word[20];
int k = 0;
while((ch = fgetc(fptr)) !=',')
{
    buf[j].word[k++] = ch;
}
buf[j].word[k] ='\0';
k = 0;
while ((ch = fgetc(fptr)) !=',')
{
    buf[j].meaning_1[k++] = ch;
}
buf[j].meaning_1[k] ='\0';
k = 0;
while ((ch = fgetc(fptr)) !='\n')
{
    if (ch == EOF)
    {
        buf[j].meaning_2[k] ='\0';
```

```
                    end = 1;
                    unlock(0);
                    return NULL;
                }
                buf[j].meaning 2[k++] = ch;
            }
            buf[j].meaning 2[k] ='\0';
            j += 1;
        }
        int k;
        i += 1;
        unlock(0);
    }
    pthread exit(0);
}
void *consumer(void *args)
{
    int i = 0;
    while (1)
    {
```

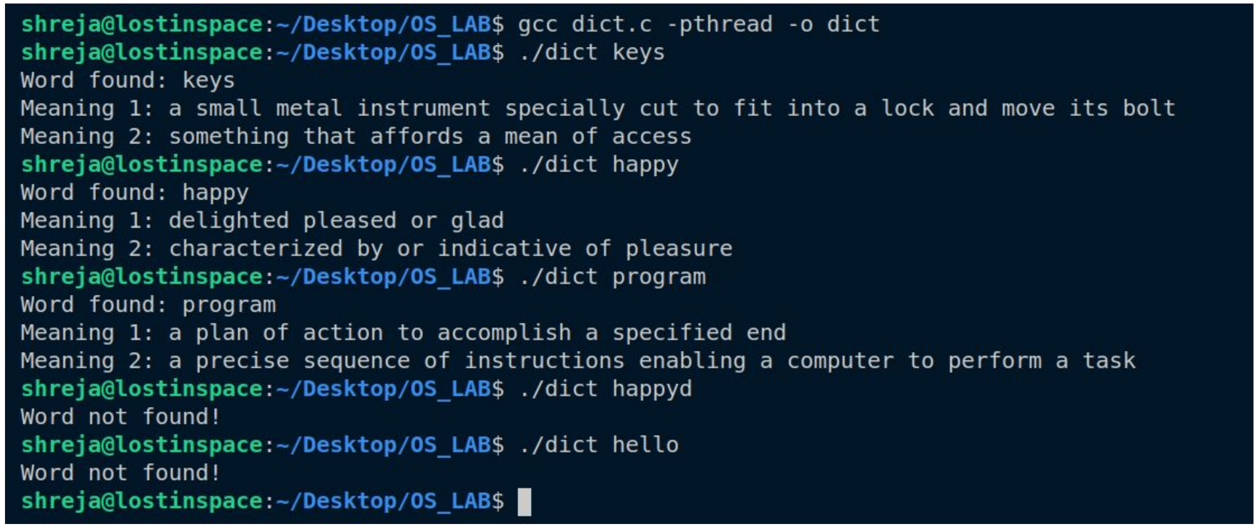
```
lock(1);
int k = 0;
while (k < j)
{
    if (strcmp(buf[k].word, search word) == 0)
    {
        printf("Word found: %s\n", buf[k].word);
        printf("Meaning 1: %s\n", buf[k].meaning 1);
        printf("Meaning 2: %s\n", buf[k].meaning 2);
        unlock(1);
        return NULL;
    }

    k += 1;
}
i += 1;
if (end == 1)
{
    printf("Word not found!\n");
    return NULL;
```

```
        }
        unlock(1);
    }
    pthread exit(0);
}
```

DICTIONARY CONTENTS:

1. happy,delighted pleased or glad,characterized by or indicative of pleasure
2. lucky,happening fortunately,bringing or foretelling good luck
3. sound,the sensation produced by stimulation of the organs of hearing by vibrations transmitted through the air or other medium,to give forth
4. bear,to hold up,to hold or remain firm under
5. bear,to hold up,to hold or remain firm under
6. dictionary,a book giving information on particular subjects or on a particular class of words names or facts usually arranged alphabetically,a list of words used by a word-processing program as the standard against which to check the spelling of text entered
7. keys,a small metal instrument specially cut to fit into a lock and move its bolt,something that affords a mean of access
8. program,a plan of action to accomplish a specified end,a precise sequence of instructions enabling a computer to perform a task
9. program,a plan of action to accomplish a specified end,a precise sequence of instructions enabling a computer to perform a task
10. program,a plan of action to accomplish a specific and precise sequence of instructions enabling a computer to perform a task
11. processes,a systematic series of actions directed to some end,the condition of being carried on
12. processes,a systematic series of actions directed to some end,the condition of being carried on
13. book,compilation of pages,read material
14. book,compilation of pages,read material
15. book,compilation of pages,read material
16. book,compilation of pages,read material
17. book,compilation of pages,read material
18. class,thing in c,where students learn
19. book,compilation of pages,read material
20. book,compilation of pages,read material

OUTPUT:



```
shreja@lostinspace:~/Desktop/OS_LAB$ gcc dict.c -pthread -o dict
shreja@lostinspace:~/Desktop/OS_LAB$ ./dict keys
Word found: keys
Meaning 1: a small metal instrument specially cut to fit into a lock and move its bolt
Meaning 2: something that affords a mean of access
shreja@lostinspace:~/Desktop/OS_LAB$ ./dict happy
Word found: happy
Meaning 1: delighted pleased or glad
Meaning 2: characterized by or indicative of pleasure
shreja@lostinspace:~/Desktop/OS_LAB$ ./dict program
Word found: program
Meaning 1: a plan of action to accomplish a specified end
Meaning 2: a precise sequence of instructions enabling a computer to perform a task
shreja@lostinspace:~/Desktop/OS_LAB$ ./dict happyd
Word not found!
shreja@lostinspace:~/Desktop/OS_LAB$ ./dict hello
Word not found!
shreja@lostinspace:~/Desktop/OS_LAB$ 
```