Operating Systems Lab

Assessment – 6

Siddhi Singh
17BIT0028

QUESTION 1

Implement the solution for reader – writer's problem.

CODE

```
#include<stdio.h>
#include<stdbool.h>
struct semaphore
{
int mutex;
int rcount;
int rwait;
bool wrt;
};
void addR(struct semaphore *s){
if (s->mutex == 0 && s->rcount == 0){
printf("\nSorry, File open in Write mode.\nNew
Reader added to queue.\n");
s->rwait++;
}
else
{
printf("\nReader Process added.\n");
s->rcount++;
s->mutex--;
}
return ;
```

```
}
void addW(struct semaphore *s)
{
if(s->mutex==1)
{
s->mutex--;
s->wrt=1;
printf("\nWriter Process added.\n");
}
else if(s->wrt) printf("\nSorry, Writer already
operational.\n");
else printf("\nSorry, File open in Read mode.\n");
return ;
}
void remR(struct semaphore *s)
{
if(s->rcount == 0) printf("\nNo readers to remove.
\n");
else
{
printf("\nReader Removed.\n");
s->rcount--;
s->mutex++;
}
return ;
}
void remW(struct semaphore *s)
{
if(s->wrt==0) printf("\nNo Writer to Remove");
else
{
```

```
printf("\nWriter Removed\n");
s->mutex++;
s->wrt=0;
if(s->rwait!=0)
{
s->mutex-=s->rwait;
s->rcount=s->rwait;
s->rwait=0;
printf("%d waiting Readers Added.",s->rcount);
}
}
}
int main()
{
struct semaphore S1={1,0,0};
while(1)
{
system("cls");
printf("Options :-\n1.Add Reader.\n2.Add Writer.
\n3.Remove Reader.\n4.Remove Writer.\n5.Exit.
\n\n\tChoice : ");
int ch;
scanf("%d",&ch);
switch(ch)
{
case 1: addR(&S1); break;
case 2: addW(&S1); break;
case 3: remR(&S1); break;
case 4: remW(&S1); break;
case 5: printf("\n\tGoodBye!"); return 0;
default: printf("\nInvalid Entry!"); continue;
```

```
}
printf("\n\n<<<<<< Current Status
>>>>>>\n\n\tMutex\t\t:\t%d\n\tActive
Readers\t:\t%d\n\tWaiting Readers\t:
\t%d\n\tWriter Active\t:\t%s\n\n", S1.mutex,
S1.rcount, S1.rwait, (S1.mutex==0 &&
S1.rcount==0) ? "YES" : "NO");
system("pause");
}
return 0;
}
```

OUTPUT

```
Options :-
1.Add Reader.
2.Add Writer.
3.Remove Reader.
4.Remove Writer.
5.Exit.

        Choice :  1

Reader Process added.


<<<<<< Current Status >>>>>>

        Mutex           :       0
        Active Readers  :       1
        Waiting Readers :       0
        Writer Active   :       NO
```

```
sh: 1: pause: not found
sh: 1: cls: not found
Options :-
1.Add Reader.
2.Add Writer.
3.Remove Reader.
4.Remove Writer.
5.Exit.

        Choice :  2

Sorry, File open in Read mode.


<<<<<< Current Status >>>>>>

        Mutex           :       0
        Active Readers  :       1
        Waiting Readers :       0
        Writer Active   :       NO
```

```
sh: 1: pause: not found
sh: 1: cls: not found
Options :-
1.Add Reader.
2.Add Writer.
3.Remove Reader.
4.Remove Writer.
5.Exit.

        Choice :  3

Reader Removed.


<<<<<< Current Status >>>>>>

        Mutex           :       1
        Active Readers  :       0
        Waiting Readers :       0
        Writer Active   :       NO
```

```
sh: 1: pause: not found
sh: 1: cls: not found
Options :-
1.Add Reader.
2.Add Writer.
3.Remove Reader.
4.Remove Writer.
5.Exit.

        Choice :  2

Writer Process added.


<<<<<< Current Status >>>>>>

        Mutex           :       0
        Active Readers  :       0
        Waiting Readers :       0
        Writer Active   :       YES
```

```
sh: 1: pause: not found
sh: 1: cls: not found
Options :-
1.Add Reader.
2.Add Writer.
3.Remove Reader.
4.Remove Writer.
5.Exit.

        Choice :  1

Sorry, File open in Write mode.
New Reader added to queue.


<<<<<< Current Status >>>>>>

        Mutex           :       0
        Active Readers  :       0
        Waiting Readers :       1
        Writer Active   :       YES
```

```
sh: 1: pause: not found
sh: 1: cls: not found
Options :-
1.Add Reader.
2.Add Writer.
3.Remove Reader.
4.Remove Writer.
5.Exit.

        Choice :  4

Writer Removed
1 waiting Readers Added.

<<<<<< Current Status >>>>>>

        Mutex           :       0
        Active Readers  :       1
        Waiting Readers :       0
        Writer Active   :       NO
```

```
sh: 1: pause: not found
sh: 1: cls: not found
Options :-
1.Add Reader.
2.Add Writer.
3.Remove Reader.
4.Remove Writer.
5.Exit.

        Choice :  5

        GoodBye!
```

QUESTION 2

Implement the solution for dining philosopher's problem.

CODE

```
#include<stdio.h>
#define n 4
int PhilosCompleted = 0,i;
struct fork{
int taken;
}ForkAvail[n];
struct philosp{
int left;
int right;
}PhilosStatus[n];

void goForDinner(int Phil_ID){
if(PhilosStatus[Phil_ID].left==10 &&
PhilosStatus[Phil_ID].right==10)
```

```
printf("Philosopher %d completed his
dinner\n",Phil_ID+1);


else if(PhilosStatus[Phil_ID].left==1 &&
PhilosStatus[Phil_ID].right==1){


printf("Philosopher %d completed his
dinner\n",Phil_ID+1);


PhilosStatus[Phil_ID].left = PhilosStatus[Phil_ID].right =
10;
int otherFork = Phil_ID-1;


if(otherFork== -1)
otherFork=(n-1);


ForkAvail[Phil_ID].taken = ForkAvail[otherFork].taken
= 0;
printf("Philosopher %d released fork %d and fork
%d\n",Phil_ID+1,Phil_ID+1,otherFork+1);
PhilosCompleted++;
}
else if(PhilosStatus[Phil_ID].left==1 &&
PhilosStatus[Phil_ID].right==0){ if(Phil_ID==(n-1)){
```

```
if(ForkAvail[Phil_ID].taken==0)
{ ForkAvail[Phil_ID].taken = PhilosStatus[Phil_ID].
right = 1;
printf("Fork %d taken by philosopher
%d\n",Phil_ID+1,Phil_ID+1);
}else{
printf("Philosopher %d is waiting for fork
%d\n",Phil_ID+1,Phil_ID+1);
}
}else{
int dupPhil_ID = Phil_ID;
Phil_ID-=1;

if(Phil_ID== -1)
Phil_ID=(n-1);
if(ForkAvail[Phil_ID].taken == 0){
ForkAvail[Phil_ID].taken =
PhilosStatus[dupPhil_ID].right = 1;
printf("Fork %d taken by Philosopher
%d\n",Phil_ID+1,dupPhil_ID+1);
}else{
printf("Philosopher %d is waiting for Fork
%d\n",dupPhil_ID+1,Phil_ID+1);
```

```
}
}
}
else if(PhilosStatus[Phil_ID].left==0){
if(Phil_ID==(n-1)){
if(ForkAvail[Phil_ID-1].taken==0){
ForkAvail[Phil_ID-1].taken = PhilosStatus[Phil_ID].left = 1;
printf("Fork %d taken by philosopher %d\n",Phil_ID,Phil_ID+1);
}else{
printf("Philosopher %d is waiting for fork %d\n",Phil_ID+1,Phil_ID);
}
}else{
if(ForkAvail[Phil_ID].taken == 0){
ForkAvail[Phil_ID].taken = PhilosStatus[Phil_ID].left = 1;
printf("Fork %d taken by Philosopher %d\n",Phil_ID+1,Phil_ID+1);
}else{
printf("Philosopher %d is waiting for Fork %d\n",Phil_ID+1,Phil_ID+1);
```

```
}
}
}else{}
}

int main(){
for(i=0;i<n;i++)
ForkAvail[i].taken=PhilosStatus[i].left=PhilosStatus[i].ri
ght=0;

while(PhilosCompleted<n){
for(i=0;i<n;i++)
goForDinner(i);
printf("\nTill now num of philosophers completed
dinner are %d\n\n",PhilosCompleted);
}

return 0;
}
```

OUTPUT

```
gcc version 4.6.3
:
Fork 1 taken by Philosopher 1
Fork 2 taken by Philosopher 2
Fork 3 taken by Philosopher 3
Philosopher 4 is waiting for fork 3

Till now num of philosophers completed dinner are 0

Fork 4 taken by Philosopher 1
Philosopher 2 is waiting for Fork 1
Philosopher 3 is waiting for Fork 2
Philosopher 4 is waiting for fork 3

Till now num of philosophers completed dinner are 0

Philosopher 1 completed his dinner
Philosopher 1 released fork 1 and fork 4
Fork 1 taken by Philosopher 2
Philosopher 3 is waiting for Fork 2
Philosopher 4 is waiting for fork 3

Till now num of philosophers completed dinner are 1

Philosopher 1 completed his dinner
Philosopher 2 completed his dinner
Philosopher 2 released fork 2 and fork 1
Fork 2 taken by Philosopher 3
Philosopher 4 is waiting for fork 3

Till now num of philosophers completed dinner are 2

Philosopher 1 completed his dinner
Philosopher 2 completed his dinner
Philosopher 3 completed his dinner
Philosopher 3 released fork 3 and fork 2
Fork 3 taken by philosopher 4

Till now num of philosophers completed dinner are 3
```

```
Philosopher 1 completed his dinner
Philosopher 2 completed his dinner
Philosopher 3 completed his dinner
Fork 4 taken by philosopher 4

Till now num of philosophers completed dinner are 3

Philosopher 1 completed his dinner
Philosopher 2 completed his dinner
Philosopher 3 completed his dinner
Philosopher 4 completed his dinner
Philosopher 4 released fork 4 and fork 3

Till now num of philosophers completed dinner are 4

: r
```

QUESTION 3

A pair of processes involved in exchanging a sequence of integers. The number of integers that can be produced and consumed at a time is limited to 100. Write a Program to implement the producer and consumer problem using POSIX semaphore for the above scenario.

CODE

```
#include<stdio.h>
#include<semaphore.h>
#include<pthread.h>
#include<stdlib.h>
#define buffersize 10
pthread_mutex_t mutex;
pthread_t tidP[20],tidC[20];
sem_t full,empty;
int counter;
int buffer[buffersize];
void initialize()
```

```
{
    pthread_mutex_init(&mutex,NULL);
    sem_init(&full,1,0);
    sem_init(&empty,1,buffersize);
    counter=0;
}
void write(int item)
{
    buffer[counter++]=item;
}
int read()
{
    return(buffer[--counter]);
}
void * producer (void * param)
{
    int waittime,item,i;
    item=rand()%5;
    waittime=rand()%5;
    sem_wait(&empty);
```

```
        pthread_mutex_lock(&mutex);
        printf("\nProducer has produced item:
%d\n",item);
        write(item);
        pthread_mutex_unlock(&mutex);
        sem_post(&full);
}
void * consumer (void * param)
{
        int waittime,item;
        waittime=rand()%5;
        sem_wait(&full);
        pthread_mutex_lock(&mutex);
        item=read();
        printf("\nConsumer has consumed item:
%d\n",item);
        pthread_mutex_unlock(&mutex);
        sem_post(&empty);
}
int main()
{
```

```
    int n1,n2,i;
    initialize();
    printf("\nEnter the no of producers: ");
    scanf("%d",&n1);
    printf("\nEnter the no of consumers: ");
    scanf("%d",&n2);
    for(i=0;i<n1;i++)

pthread_create(&tidP[i],NULL,producer,NULL);
    for(i=0;i<n2;i++)

pthread_create(&tidC[i],NULL,consumer,NULL)
;
    for(i=0;i<n1;i++)
        pthread_join(tidP[i],NULL);
    for(i=0;i<n2;i++)
        pthread_join(tidC[i],NULL);
    sleep(5);
    exit(0);
}
```

OUTPUT

```
gcc version 4.6.3
main.c: In function 'main':
main.c:65:2: warning: implicit declaration of function 'sleep' [-Wimplicit-function-declaration]
  sleep(5);
  ^~~~~

Enter the no of producers:  4

Enter the no of consumers:  4

Producer has produced item: 3

Consumer has consumed item: 3

Producer has produced item: 1

Consumer has consumed item: 1

Producer has produced item: 4

Consumer has consumed item: 4

Producer has produced item: 2

Consumer has consumed item: 2
```