21BAI1217
Mainak Chattopadhyay
L27+L28

OS LAB 8

1. Write a C program to implement producer consumer problem using semaphore.

CODE -

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

int mutex=1,full=0,empty=3,x=0;

int main()
{
        int n;
        void producer();
        void consumer();
        int wait(int);
        int signal(int);
        printf("\n1.Producer\n2.Consumer\n3.Exit");
        while(1)
        {
                printf("\nEnter your choice:");
                scanf("%d",&n);
                switch(n)
                {
                        case 1:if((mutex==1)&&(empty!=0))
                                        producer();
                                else
                                        printf("Buffer is full!!");
                                break;
                        case 2:if((mutex==1)&&(full!=0))
                                        consumer();
                                else
                                        printf("Buffer is empty!!");
                                break;
                        case 3:
                                exit(0);
                                break;
                }
        }
```

```
        return 0;
}

int wait(int s)
{
        return (--s);
}

int signal(int s)
{
        return(++s);
}

void producer()
{
        mutex=wait(mutex);
        full=signal(full);
        empty=wait(empty);
        x++;
        printf("\nProducer produces the item %d",x);
        mutex=signal(mutex);
}

void consumer()
{
        mutex=wait(mutex);
        full=wait(full);
        empty=signal(empty);
        printf("\nConsumer consumes item %d",x);
        x--;
        mutex=signal(mutex);
}
```

OUTPUT-


ex2@AB1205BSCS022: ~
File Edit View Search Terminal Help
ex2@AB1205BSCS022:~$ gedit lab8q1.c
ex2@AB1205BSCS022:~$ gcc lab8q1.c
ex2@AB1205BSCS022:~$ ./a.out
1.Producer
2.Consumer
3.Exit
Enter your choice:1
Producer produces the item 1
Enter your choice:2
Consumer consumes item 1
Enter your choice:2
Buffer is empty!!
Enter your choice:3
ex2@AB1205BSCS022:~$

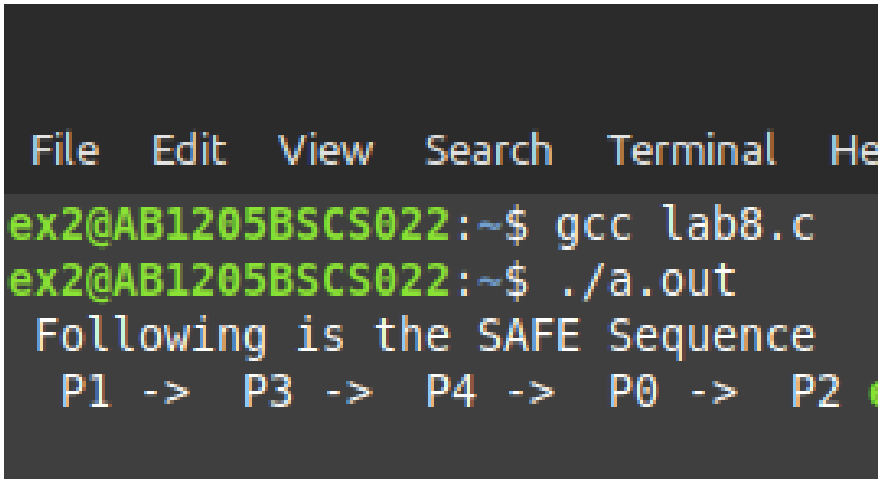
2. Write a C program to implement deadlock avoidance by using Banker's algorithm.

CODE -

```
#include<stdio.h>
int main()
{
   // P0 , P1 , P2 , P3 , P4 are the Process names here
   int n , m , i , j , k;
   n = 5; // Number of processes
   m = 3; // Number of resources
   int alloc[ 5 ] [ 3 ] = { { 0 , 1 , 0 }, // P0 // Allocation Matrix
                    { 2 , 0 , 0 } , // P1
                    { 3 , 0 , 2 } , // P2
                    { 2 , 1 , 1 } , // P3
                    { 0 , 0 , 2 } } ; // P4
   int max[ 5 ] [ 3 ] = { { 7 , 5 , 3 } , // P0 // MAX Matrix
                { 3 , 2 , 2 } , // P1
                { 9 , 0 , 2 } , // P2
                { 2 , 2 , 2 } , // P3
                { 4 , 3 , 3 } } ; // P4
   int avail[3] = { 3 , 3 , 2 } ; // Available Resources
   int f[n] , ans[n] , ind = 0 ;
   for (k = 0; k < n; k++) {
      f[k] = 0;
   }
   int need[n][m];
   for (i = 0; i < n; i++) {
      for (j = 0; j < m; j++)
         need[i][j] = max[i][j] - alloc[i][j] ;
   }
   int y = 0;
   for (k = 0; k < 5; k++){
      for (i = 0; i < n; i++){
         if (f[i] == 0){
            int flag = 0;
            for (j = 0; j < m; j++) {
               if(need[i][j] > avail[j]){
                  flag = 1;
                  break;
               }
```

```
            }
            if ( flag == 0 ) {
                ans[ind++] = i;
                for (y = 0; y < m; y++)
                    avail[y] += alloc[i][y] ;
                f[i] = 1;
            }
          }
        }
    }
    int flag = 1;
    for(int i=0;i<n;i++)
    {
    if(f[i] == 0)
    {
        flag = 0;
        printf(" The following system is not safe ");
        break;
    }
    }
    if (flag == 1)
    {
    printf(" Following is the SAFE Sequence \ n ");
    for (i = 0; i < n - 1; i++)
        printf(" P%d -> " , ans[i]);
    printf(" P%d ", ans[n - 1]);
    }
    return(0);
}
```

OUTPUT-


File Edit View Search Terminal He
ex2@AB1205BSCS022:~$ gcc lab8.c
ex2@AB1205BSCS022:~$ ./a.out
Following is the SAFE Sequence
P1 -> P3 -> P4 -> P0 -> P2