Lab # 11

Deadlock, Deadlock Avoidance and Detection

In this lab we will look at the following points

i. **Example where Deadlock Occurs**
ii. **Banker's Algorithm for Deadlock Avoidance**
iii. **Deadlock Detection Algorithm**

Deadlock

Example # 01

```
#include<pthread.h>
#include<stdlib.h>
#include<unistd.h>
#include<stdio.h>
#include <stdbool.h>
bool lock1 = false;
bool lock2=false;
int myGlobal = 0;
void *threadFunction1()
{
int i, j;
for (i = 0; i<10; i++)
{
  printf("\n I am thread # 01 \n");
while(lock1);
lock1=true;
j = myGlobal;
j = j+1;
myGlobal = j;
sleep(1);
while(lock2);
lock2=true;
printf("\n My Global Is: %d\n", myGlobal);
lock2=false;
lock1=false;
}
}
void *threadFunction2()
{
int i, j;
for (i = 0; i<10; i++)
```

```
{
  printf("\n I am thread # 02 \n");
while(lock2);
lock2=true;
j = myGlobal;
j = j+1;
myGlobal = j;
while(lock1);
lock1=true;
printf("\n My Global Is: %d\n", myGlobal);
lock1=false;
lock2=false;
}
}
int main()
{
pthread_t myThread1,myThread2;
int i,k;
pthread_create(&myThread1, NULL,threadFunction1,NULL);
pthread_create(&myThread2, NULL,threadFunction2,NULL);
pthread_join(myThread1, NULL);
pthread_join(myThread2, NULL);
exit(0);
}
```

Banker's Algorithm:

Example # 02

```
#include<stdio.h>
struct file
{
int all[10];
int max[10];
int need[10];
int flag;
};
int main()
{
struct file f[10];
int fl;
int i, j, k, p, b, n, r, g, cnt=0, id, newr;
int avail[10],seq[10];
printf("Enter number of processes -- ");
scanf("%d",&n);
printf("Enter number of resources -- ");
scanf("%d",&r);
for(i=0;i<n;i++)
{
printf("Enter details for P%d",i);
printf("\nEnter allocation\t -- \t");
```

```
for(j=0;j<r;j++)
scanf("%d",&f[i].all[j]);
printf("Enter Max\t\t -- \t");
for(j=0;j<r;j++)
scanf("%d",&f[i].max[j]);
f[i].flag=0;
}
printf("\nEnter Available Resources\t -- \t");
for(i=0;i<r;i++)
scanf("%d",&avail[i]);
printf("\nEnter New Request Details -- ");
printf("\nEnter pid \t -- \t");
scanf("%d",&id);
printf("Enter Request for Resources \t -- \t");
for(i=0;i<r;i++)
{
scanf("%d",&newr);
f[id].all[i] += newr;
avail[i]=avail[i] - newr;
}
for(i=0;i<n;i++)
{
for(j=0;j<r;j++)
{
f[i].need[j]=f[i].max[j]-f[i].all[j];
if(f[i].need[j]<0)
f[i].need[j]=0;
}
}
cnt=0;
fl=0;
while(cnt!=n)
{
g=0;
for(j=0;j<n;j++)
{
if(f[j].flag==0)
{
b=0;
for(p=0;p<r;p++)
{
if(avail[p]>=f[j].need[p])
b=b+1;
else
b=b-1;
}
if(b==r)
{
printf("\nP%d is visited",j);
seq[fl++]=j;
f[j].flag=1;
for(k=0;k<r;k++)
```

```
avail[k]=avail[k]+f[j].all[k];
cnt=cnt+1;
printf("(");
for(k=0;k<r;k++)
printf("%3d",avail[k]);
printf(")");
g=1;
}
}
}
if(g==0)
{
printf("\n REQUEST NOT GRANTED -- DEADLOCK OCCURRED");
printf("\n SYSTEM IS IN UNSAFE STATE");
goto y;
}
}
printf("\nSYSTEM IS IN SAFE STATE");
printf("\nThe Safe Sequence is -- (");
for(i=0;i<fl;i++)
printf("P%d ",seq[i]);
printf(")");
y: printf("\nProcess\t\tAllocation\t\tMax\t\t\tNeed\n");
for(i=0;i<n;i++)
{
printf("P%d\t",i);
for(j=0;j<r;j++)
printf("%6d",f[i].all[j]);
for(j=0;j<r;j++)
printf("%6d",f[i].max[j]);
for(j=0;j<r;j++)
printf("%6d",f[i].need[j]);
printf("\n");
}
return 0;
}
```

Deadlock Avoidance Algorithm:

Example # 03

```
#include<stdio.h>
int max[100][100];
int alloc[100][100];
int need[100][100];
int avail[100];
int n,r;
void input();
void show();
void cal();
int main()
```

```
{
int i,j;
  printf("********** Deadlock Detection Algo ************\n");
  input();
  show();
cal();
  return 0;
}
void input()
{
  int i,j;
  printf("Enter the no of Processes\t");
  scanf("%d",&n);
printf("Enter the no of resource instances\t");
scanf("%d",&r);
printf("Enter the Max Matrix\n");
for(i=0;i<n;i++)
  {
    for(j=0;j<r;j++)
    {
      scanf("%d",&max[i][j]);
    }
  }
printf("Enter the Allocation Matrix\n");
for(i=0;i<n;i++)
{
    for(j=0;j<r;j++)
    {
      scanf("%d",&alloc[i][j]);
    }
  }
  printf("Enter the available Resources\n");
  for(j=0;j<r;j++)
  {
    scanf("%d",&avail[j]);
  }
}
void show()
{
int i,j;
printf("Process\t Allocation\t Max\t Available\t");
  for(i=0;i<n;i++)
  {
  printf("\nP%d\t   ",i);
    for(j=0;j<r;j++)
    {
      printf("%d ",alloc[i][j]);
    }
    printf("\t");
    for(j=0;j<r;j++)
    {
      printf("%d ",max[i][j]);
```

```
    }
    printf("\t");
    if(i==0)
    {
      for(j=0;j<r;j++)
      printf("%d ",avail[j]);
    }
  }
}
void cal()
{
int finish[100],temp,need[100][100],flag=1,k,c1=0;
int dead[100];
  int safe[100];
  int i,j;
  for(i=0;i<n;i++)
  {
    finish[i]=0;
  }
  //find need matrix
  for(i=0;i<n;i++)
  {
    for(j=0;j<r;j++)
    {
      need[i][j]=max[i][j]-alloc[i][j];
    }
}
while(flag)
{
    flag=0;
     for(i=0;i<n;i++)
    {
      int c=0;
      for(j=0;j<r;j++)
      {
        if((finish[i]==0)&&(need[i][j]<=avail[j]))
        {
           c++;
          if(c==r)
          {
            for(k=0;k<r;k++)
            {
              avail[k]+=alloc[i][j];
              finish[i]=1;
              flag=1;
                  }
            //printf("\nP%d",i);
            if(finish[i]==1)
            {
              i=n;
          }
                }
```

```
        }
      }
      }
   }
   j=0;
   flag=0;
   for(i=0;i<n;i++)
   {
    if(finish[i]==0)
    {
      dead[j]=i;
      j++;
      flag=1;
    }
  }
  if(flag==1)
  {
    printf("\n\nSystem is in Deadlock and the Deadlock process are\n");
    for(i=0;i<j;i++)
    {
      printf("P%d\t",dead[i]);
    }
}
  else
  {
    printf("\nNo Deadlock Occur");
  }
}
```