DEADLOCK DETECTION:

kaushik@kaushik-AcerPower-Series:~/OS $ cat deadlockdetec.c

```
#include<stdio.h>
#define RELEASE 0
#define ASSIGN 1
#define REQUEST 2
#define WAIT 1
#define FALSE 0
#define TRUE 1
void Initialise(int a[][10],int n,int m)
{
int i,j;
for(i=0;i<n;i++)
for(j=0;j<m;j++)
a[i][j]=0;
return;
}
void DeadlockPath(int pred[][10],int i,int j)
{
if(pred[i][j]==-1)
{ printf("\b\bP%d->P%d",i,j);
return;
} else
{ DeadlockPath(pred,i,pred[i][j]);
DeadlockPath(pred,pred[i][j],j);
} return;
}
void Detection(int Wait[][10],int n)
{
int i,j,k,pred[10][10],graph[10][10],flag=FALSE;
for(i=0;i<n;i++)
for(j=0;j<n;j++)
{ pred[i][j]=-1;
graph[i][j]=Wait[i][j];
}
for(k=0;k<n;k++)
for(i=0;i<n;i++)
for(j=0;j<n;j++)
if(graph[i][j]!=1)
if(graph[i][k]==1&&graph[k][j]==1)
{ pred[i][j]=k;
graph[i][j]=1;
}
for(i=0;i<n;i++)
if(pred[i][i]!=-1)
{ printf("\nDeadlock Detected: P%d",i);
DeadlockPath(pred,i,pred[i][i]);
DeadlockPath(pred,pred[i][i],i);
flag=TRUE;
}
if(flag==FALSE) printf("\nNo deadlock has been detected\n");
else printf("\n");
return;
}
void Display(int a[][10],int n,int m)
```

```
{
int i,j;
printf("\n");
for(i=0;i<n;i++)
{ printf("P%d",i);
for(j=0;j<m;j++)
printf("\t%d",a[i][j]);
printf("\n");
} return;
}
void WaitGraph(int Wait[][10],int Resource[][10],int nR,int nP)
{
Initialise(Wait,nP,nP);
int i,j,k;
for(i=0;i<nP;i++)
for(j=0;j<nR;j++)
if(Resource[i][j]==REQUEST)
for(k=0;k<nP;k++)
if(Resource[k][j]==ASSIGN)
Wait[i][k]=WAIT;
return;
}
void main()
{
int i,d,nR,nP,pid,rid,flag,Resource[10][10],Wait[10][10],choice=-1;
printf("\nDEADLOCK DETECTION\n");
printf("Enter the number of resources: ");
scanf("%d",&nR);
printf("Enter the number of processes: ");
scanf("%d",&nP);
Initialise(Resource,nP,nR);
Initialise(Wait,nP,nR);
printf("\n1. REQUEST\n2. RELEASE\n3. DEADLOCK DETECTION\n4. RESOURCE
ALLOCATION GRAPH\n5. WAIT FOR GRAPH\n6. EXIT");
while(choice!=6)
{
printf("\nEnter an option: ");
scanf("%d",&choice);
switch(choice)
{
case 1://REQUEST
printf("REQUEST: ");
printf("Enter the Process and Resource ID: ");
scanf("%d%d",&pid,&rid);
flag=FALSE;
for(i=0;i<nP;i++)
if(Resource[i][rid]==ASSIGN)
{ flag=TRUE;
break;
}
if(flag==FALSE)
{ printf("Process P%d has been assigned the Resource R%d\n",pid,rid);
Resource[pid][rid]=ASSIGN;
} else
{ printf("Process P%d is waiting for the Resource R%d\n",pid,rid);
Resource[pid][rid]=REQUEST;
}
```

```
break;
case 2://RELEASE
printf("RELEASE: ");
printf("Enter the Process and Resource ID: ");
scanf("%d",&pid);
printf("Enter the Resource ID: ");
scanf("%d",&rid);
Resource[pid][rid]=RELEASE;
printf("Resource R%d has been released from the process P%d\n",rid,pid);
for(i=0;i<nP;i++)
if(Resource[i][rid]==REQUEST)
{
Resource[i][rid]=ASSIGN;
printf("Process P%d has been assigned the Resource R%d\n",i,rid);
break;
}
break;
case 3://DEADLOCK DETECTION
printf("DEADLOCK DETECTION");
WaitGraph(Wait,Resource,nR,nP);
Detection(Wait,nP);
break;
case 4://PRINT RESOURCE ALLOCATION GRAPH
printf("RESOURCE ALLOCATION GRAPH\n");
printf("ASSIGN=%d REQUEST=%d RELEASE=%d\n\n",ASSIGN,REQUEST,RELEASE);
for(i=0;i<nR;i++)
printf("\tR%d",i);
Display(Resource,nP,nR);
break;
case 5://PRINT WAIT FOR GRAPH
printf("\nWAIT FOR GRAPH\nWAIT=%d\n\n",WAIT);
WaitGraph(Wait,Resource,nR,nP);
for(i=0;i<nP;i++)
printf("\tP%d",i);
Display(Wait,nP,nP);
break;
case 6:
break;
}
}
}
kaushik@kaushik-AcerPower-Series:~/OS $
kaushik@kaushik-AcerPower-Series:~/OS $ gcc -w -o a  deadlockdetec.c
kaushik@kaushik-AcerPower-Series:~/OS $ ./a
DEADLOCK DETECTION
Enter the number of resources: 5
Enter the number of processes: 5
1. REQUEST
2. RELEASE
3. DEADLOCK DETECTION
4. RESOURCE ALLOCATION GRAPH
5. WAIT FOR GRAPH
6. EXIT
Enter an option: 1
REQUEST: Enter the Process and Resource ID: 1 0
Process P1 has been assigned the Resource R0
Enter an option: 1
```

REQUEST: Enter the Process and Resource ID: 0 0
Process P0 is waiting for the Resource R0
Enter an option: 1
REQUEST: Enter the Process and Resource ID: 0 1
Process P0 has been assigned the Resource R1
Enter an option: 1
REQUEST: Enter the Process and Resource ID: 3 1
Process P3 is waiting for the Resource R1
Enter an option: 1
REQUEST: Enter the Process and Resource ID: 4 2
Process P4 has been assigned the Resource R2
Enter an option: 1
REQUEST: Enter the Process and Resource ID: 1 2
Process P1 is waiting for the Resource R2
Enter an option: 1
REQUEST: Enter the Process and Resource ID: 2 3
Process P2 has been assigned the Resource R3
Enter an option: 1
REQUEST: Enter the Process and Resource ID: 1 3
Process P1 is waiting for the Resource R3
Enter an option: 1
REQUEST: Enter the Process and Resource ID: 3 4
Process P3 has been assigned the Resource R4
Enter an option: 1
REQUEST: Enter the Process and Resource ID: 1 4
Process P1 is waiting for the Resource R4
Enter an option: 1
REQUEST: Enter the Process and Resource ID: 2 4
Process P2 is waiting for the Resource R4
Enter an option: 4
RESOURCE ALLOCATION GRAPH
ASSIGN=1 REQUEST=2 RELEASE=0
R0 R1 R2 R3 R4
P0 2 1 0 0 0
P1 1 0 2 2 2
P2 0 0 0 1 2
P3 0 2 0 0 1
P4 0 0 1 0 0
Enter an option: 5
WAIT FOR GRAPH
WAIT=1
P0 P1 P2 P3 P4
P0 0 1 0 0 0
P1 0 0 1 1 1
P2 0 0 0 1 0
P3 1 0 0 0 0
P4 0 0 0 0 0
Enter an option: 3
DEADLOCK DETECTION
Deadlock Detected: P0->P1->P3->P0
Deadlock Detected: P1->P3->P0->P1
Deadlock Detected: P2->P3->P0->P1->P2
Deadlock Detected: P3->P0->P1->P3
Enter an option: 2
RELEASE: Enter the Process and Resource ID: 3 4
Enter the Resource ID: Resource R4 has been released from the process P3
Process P1 has been assigned the Resource R4

```
Enter an option: 3
DEADLOCK DETECTION
Deadlock Detected: P1->P2->P1
Deadlock Detected: P2->P1->P2
Enter an option: 2
RELEASE: Enter the Process and Resource ID: 1 4
Enter the Resource ID: Resource R4 has been released from the process P1
Process P2 has been assigned the Resource R4
Enter an option: 3
DEADLOCK DETECTION
No deadlock has been detected
Enter an option: 6
```