Muchamad Rif’an

17.01.53.2021

Teknik Informatika R2

EXPERIMENT 8

Objective

**Write a C program to simulate Bankers algorithm for the purpose of deadlock avoidance.

Description

In a multiprogramming environment, several processes may compete for a finite number of resources. A process requests resources; if the resources are not available at that time, the process enters a waiting state. Sometimes, a waiting process is never again able to change state, because the resources it has requested are held by other waiting processes. This situation is called a deadlock. Deadlock avoidance is one of the techniques for handling deadlocks. This approach requires that the operating system be given in advance additional information concerning which resources a process will request and use during its lifetime. With this additional knowledge, it can decide for each request whether or not the process should wait. To decide whether the current request can be satisfied or must be delayed, the system must consider the resources currently available, the resources currently allocated to each process, and the future requests and releases of each process. Banker’s algorithm is a deadlock avoidance algorithm that is applicable to a system with multiple instances of each resource type.

Program

rifandeadlock.c

```
#include<stdio.h>
struct file
{
int all[10];
int max[10];
int need[10];
int flag;
};
void 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 -- ");
```

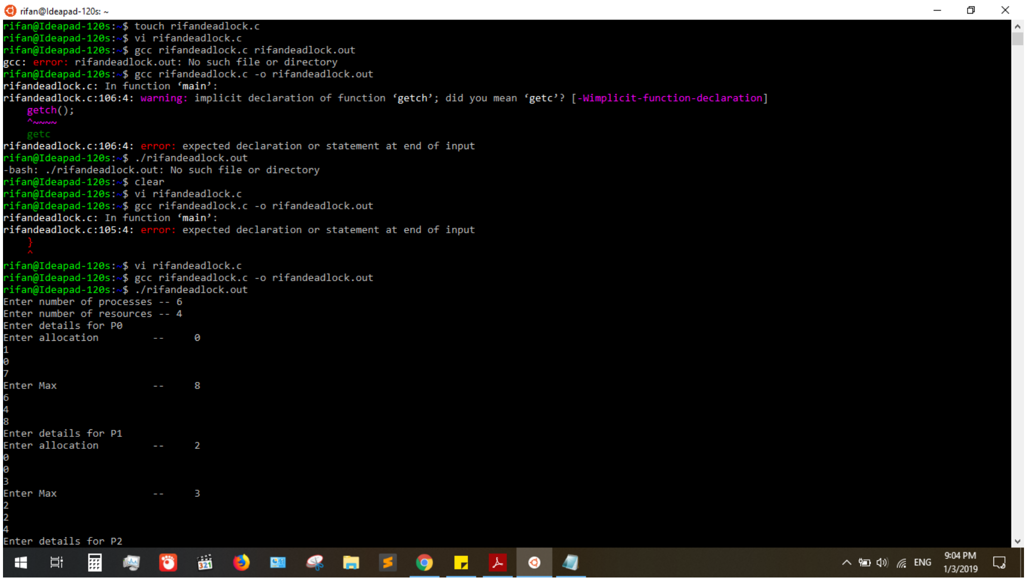
```
scanf("%d",&r);
for(i=0;i<n;i++)
{
printf("Enter details for P%d",i);
printf("\nEnter allocation\t -- \t");
for(j=0;j<r;j++)
scanf("%d",&f[i].all[j]);
printf("Enter Max\t\t -- \t");
for(j=0;j<r;j++)
scanf("%d",&f[i].max[j]);
f[i].flag=0;
}
printf("\nEnter Available Resources\t -- \t");
for(i=0;i<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");
}
}
```

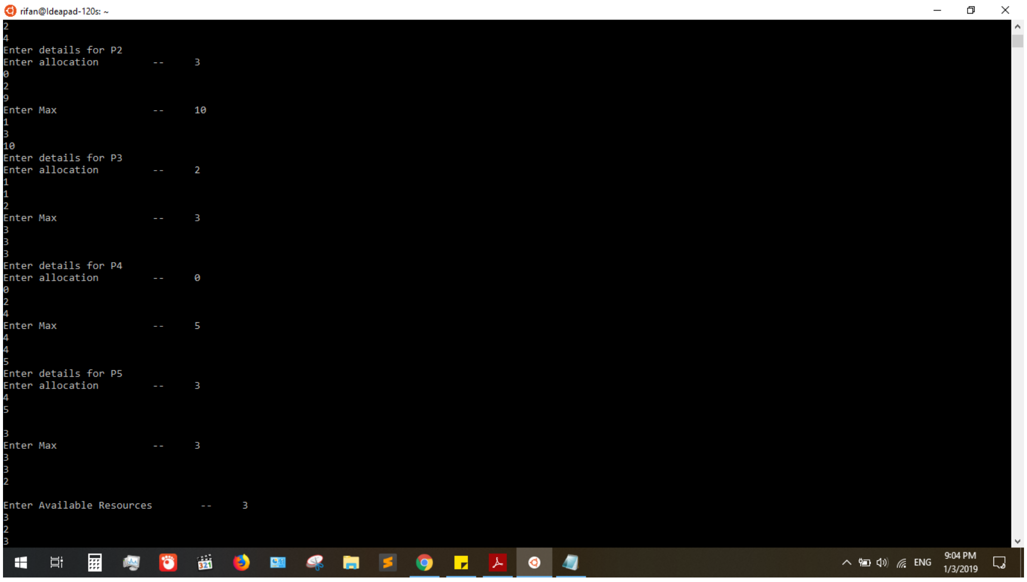
INPUT

```
Enter the number of processes :  6

Enter the number of files : 4

Enter details for P0

Enter allocation        : 0     1       0       7

Enter max               : 8     6       4       8

Enter details for P1    : 2     0       0       3

Enter max               : 3     2       2       4

Enter details for P2    : 3     0       2       9

Enter max               : 5     4       4       5

Enter details for P3    : 2     1       1       2

Enter max               : 3     3       3       3

Enter details for P4    : 0     5       2       4

Enter max               : 5     4       4       5

Enter details for P5    : 3     4       5       3

Enter max               : 3     3       3       2


Enter Avaible Resource          : 3     3       2       3

Enter new request details       --

Enter pid                       : 1

Enter request for resources     : 1     0       2       3


P1 is visited   (       5       3       2       6       )

P3 is visited    (       7      4       3       8       )

P4 is visited    (      7       4       5       12      )

P5 is visited    (      10      8       10      15      )

P0 is visited    (      10      9       10      22      )

P2 is visited   (       13      9       12      31      )

System is in safe state

The safe sequence is – (P1      P3      P4      P5      P0      P2 )
```

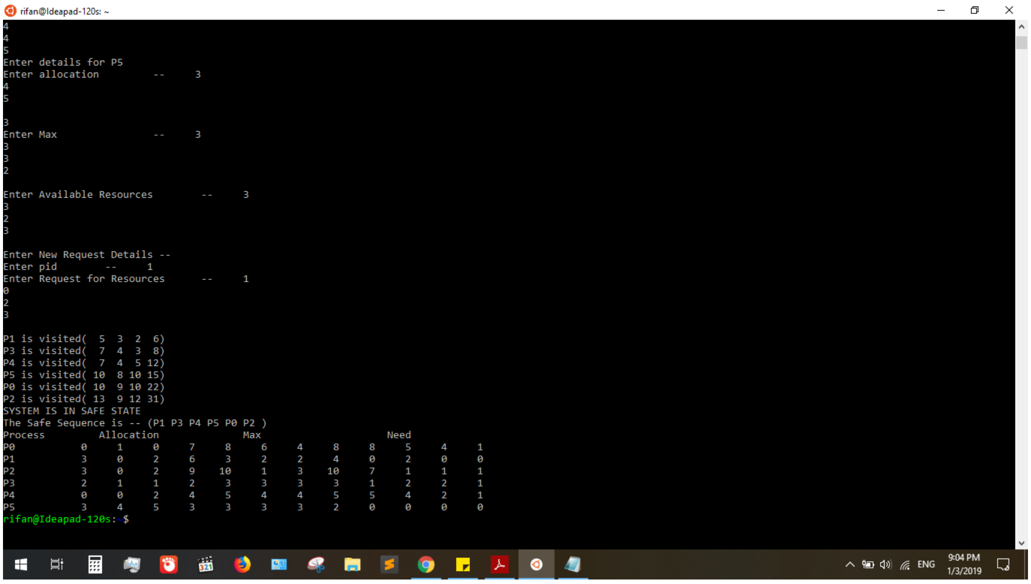
Process	Allocation				Max				Need			
P0	0	1	0	7	8	6	4	8	8	5	4	1
P1	3	0	2	6	3	2	2	4	0	2	0	0
P2	3	0	2	9	10	1	3	10	7	1	1	1
P3	2	1	1	2	3	3	3	3	1	2	2	1
P4	0	0	2	4	5	4	4	5	5	4	2	1
P5	3	4	5	3	3	3	3	2	0	0	0	0



```
rifan@Ideapad-120s:~$ touch rifandeadlock.c
rifan@Ideapad-120s:~$ vi rifandeadlock.c
rifan@Ideapad-120s:~$ gcc rifandeadlock.c rifandeadlock.out
gcc: error: rifandeadlock.out: No such file or directory
rifan@Ideapad-120s:~$ gcc rifandeadlock.c -o rifandeadlock.out
rifandeadlock.c: In function ‘main’:
rifandeadlock.c:106:4: warning: implicit declaration of function ‘getch’; did you mean ‘getc’? [-Wimplicit-function-declaration]
    getch();
    ^~~~~
    getc
rifandeadlock.c:106:4: error: expected declaration or statement at end of input
rifan@Ideapad-120s:~$ ./rifandeadlock.out
-bash: ./rifandeadlock.out: No such file or directory
rifan@Ideapad-120s:~$ clear
rifan@Ideapad-120s:~$ vi rifandeadlock.c
rifan@Ideapad-120s:~$ gcc rifandeadlock.c -o rifandeadlock.out
rifandeadlock.c: In function ‘main’:
rifandeadlock.c:105:4: error: expected declaration or statement at end of input
    }
    ^

rifan@Ideapad-120s:~$ vi rifandeadlock.c
rifan@Ideapad-120s:~$ gcc rifandeadlock.c -o rifandeadlock.out
rifan@Ideapad-120s:~$ ./rifandeadlock.out
Enter number of processes -- 6
Enter number of resources -- 4
Enter details for P0
Enter allocation         --     0
1
0
7
Enter Max                --     8
6
4
8
Enter details for P1
Enter allocation         --     2
0
0
3
Enter Max                --     3
2
2
4
Enter details for P2
```



```
2
4
Enter details for P2
Enter allocation         --     3
0
2
9
Enter Max                --     10
1
3
10
Enter details for P3
Enter allocation         --     2
1
1
2
Enter Max                --     3
3
3
3
Enter details for P4
Enter allocation         --     0
0
2
4
Enter Max                --     5
4
4
5
Enter details for P5
Enter allocation         --     3
4
5

3
Enter Max                --     3
3
3
2

Enter Available Resources        --     3
3
2
3
```



```
4
4
5
Enter details for P5
Enter allocation         --     3
4
5

3
Enter Max                --     3
3
3
2

Enter Available Resources        --     3
3
2
3

Enter New Request Details --
Enter pid        --     1
Enter Request for Resources      --     1
0
2
3

P1 is visited(  5  3  2  6)
P3 is visited(  7  4  3  8)
P4 is visited(  7  4  5 12)
P5 is visited( 10  8 10 15)
P0 is visited( 10  9 10 22)
P2 is visited( 13  9 12 31)
SYSTEM IS IN SAFE STATE
The Safe Sequence is -- (P1 P3 P4 P5 P0 P2 )
Process         Allocation              Max                     Need
P0           0     1     0     7     8     6     4     8     8     5     4     1
P1           3     0     2     6     3     2     2     4     0     2     0     0
P2           3     0     2     9    10     1     3    10     7     1     1     1
P3           2     1     1     2     3     3     3     3     1     2     2     1
P4           0     0     2     4     5     4     4     5     5     4     2     1
P5           3     4     5     3     3     3     3     2     0     0     0     0
rifan@Ideapad-120s:~$
```