**海南大学网络空间学院**

**实 验 报 告**

**实验课程： 《操作系统原理及安全》实验**

**实验名称：** 实验六

**学号：** 20213006839  **姓名：甄五四**

**专业班级： 信息安全（密码学方向）理科实验班指导教师：**秦小立

**完成日期：** 2023 年 5 月 31 日

**成绩：**

**评阅：**

**教师签名：**

1. **实验目的**
2. 理解银行家算法。
3. 掌握进程安全性检查的方法与资源分配的方法。
4. **实验任务**

编制模拟银行家算法的程序，并以下面给出的例子验证所编写的程序的正确性。

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 进程 | 已占资源 | | | | 最大需求数 | | | |
| A | B | C | D | A | B | C | D |
| P0 | 0 | 0 | 1 | 2 | 0 | 0 | 1 | 2 |
| P1 | 1 | 0 | 0 | 0 | 1 | 7 | 5 | 0 |
| P2 | 1 | 3 | 5 | 4 | 2 | 3 | 5 | 6 |
| P3 | 0 | 6 | 3 | 2 | 0 | 6 | 5 | 2 |
| P4 | 0 | 0 | 1 | 4 | 0 | 6 | 5 | 6 |

现在系统中A、B、C、D 4类资源分别还剩1、5、2、0个，请按银行家算法回答：

1. 现在系统是否处于安全状态？
2. 如果现在进程P1提出需要（0、4、2、0）个资源的请求，系统能否满足它的请求？
3. **实验环境**

VMware15.5 PRO + Ubuntu 20.04

1. **实验内容及步骤(题目、相应程序代码、运行结果)**

**题目：**编制模拟银行家算法的程序，并以给出的例子验证所编写的程序的正确性。

**程序代码：**

#include <stdio.h>

#define M 100

#define N 50 //全局变量定义

int Available[M]; //可利用资源数组

int Max[N][M]; //最大需求矩阵

int Allocation[N][M]; //分配矩阵

int Need[N][M]; //需求矩阵

int Request[N][M]; //M个进程还需要N类资源的资源量

int Finish[N];

int p[N];

int m,n; //M个进程,N类资源

//安全性算法

int Safe()

{

int i,j,k,l=0;

int Work[M]; //可利用资源数组

for (i=0;i<n;i++)

Work[i]=Available[i];

for (i=0;i<m;i++)

Finish[i]=0;

for (i=0;i<m;i++)

{

if (Finish[i]==1)

continue;

else

{

for (j=0;j<n;j++)

{

if (Need[i][j]>Work[j])

break;

}

if (j==n)

{

Finish[i]=1;

for(k=0;k<n;k++)

Work[k]+=Allocation[i][k];

p[l++]=i;

i=-1;

}

else continue;

}

if (l==m)

{

printf("系统是安全的\n");

printf("系统安全序列是:\n");

for (i=0;i<l;i++)

{

printf("%d",p[i]);

if (i!=l-1)

printf("-->");

}

printf("\n");

return 1;

}

}

return 0;

}

//银行家算法

int main()

{

int i,j,mi;

printf("请输入进程的数目:\n");

scanf("%d",&m);

printf("请输入资源的种类:\n");

scanf("%d",&n);

printf("输入每个进程对资源的最大需求量,按照%d\*%d矩阵输入\n",m,n);

for (i=0;i<m;i++)

for(j=0;j<n;j++)

scanf("%d",&Max[i][j]);

printf("输入每个进程资源的目前占有量,按照%d\*%d矩阵输入\n",m,n);

for (i=0;i<m;i++)

{

for(j=0;j<n;j++)

{

scanf("%d",&Allocation[i][j]);

Need[i][j]=Max[i][j]-Allocation[i][j];

if (Need[i][j]<0)

{

printf("你输入的第%d个进程所拥有的第%d个资源错误请重新输入:\n",i+1,j+1);

j--;

continue;

}

}

}

printf("请输入系统剩余资源量:\n");

for (i=0;i<n;i++)

scanf("%d",&Available[i]);

Safe();

while (1)

{

printf("输入要申请的资源的进程号（第一个进程号为0、第二个进程号为1，依此类推）\n"); scanf("%d",&mi);

printf("输入进程所请求的各个资源的数量\n");

for (i=0;i<n;i++)

{

scanf("%d",& Request[mi][i]);

}

for (i=0;i<n;i++)

{

if (Request[mi][i]>Need[mi][i])

{

printf("所请求资源数超过进程的需求量!\n");

return 0;

}

if (Request[mi][i]>Available[i])

{

printf("所请求资源数超过系统所有的资源数!\n");

return 0;

}

}

for (i=0;i<n;i++)

{

Available[i]-=Request[mi][i];

Allocation[mi][i]+=Request[mi][i];

Need[mi][i]-=Request[mi][i];

}

if (Safe())

printf("同意分配请求\n");

else

{

printf("SORRY你的请求被拒绝\n");

for (i=0;i<n;i++)

{

Available[i]+=Request[mi][i];

Allocation[mi][i]-=Request[mi][i];

Need[mi][i]+=Request[mi][i];

}

}

for (i=0;i<m;i++)

Finish[i]=0;

char Flag; //标志位

printf("是否再次请求分配是请按Y/y否请按N/n");

while (1)

{

scanf("%c",&Flag);

if (Flag=='Y'||Flag=='y'||Flag=='N'||Flag=='n')

break;

else

{

printf("请按要求重新输入:\n");

continue;

}

}

if (Flag=='Y'||Flag=='y')

continue;

else break;

}

return 0;

}

**运行结果：**

系统处于安全状态，并且能满足进程1的资源请求

****

**五、实验总结**

通过本次实现深入学习银行家算法，分配资源之前，判断系统是否是安全的；若是，才分配。每分配一次资源就测试一次是否安全，若不是，资源全部就位后才测试，还掌握进程安全性检查的方法与资源分配的方法。程序主函数首先从用户输入中读取进程数目m，资源种类数目n，每个进程对资源的最大需求量Max、资源已分配量Allocation、剩余资源量Available等信息，并调用Safe()函数检查当前系统是否处于安全状态。然后根据用户输入的读进程号mi，以及进程所请求的各种资源数量Request判断如果进程请求的资源数量超过了其所需要的资源数量或系统中所有资源数量，则返回错误信息，结束当前循环。否则，程序将从系统剩余资源中分配所需资源，并更新该进程的已分配资源量Allocation和需求量Need。随后，程序通过调用Safe()函数检查当前系统是否处于安全状态，如果处于安全状态，则同意分配请求；否则拒绝分配请求，并且返回分配前的系统状态。Safe()函数遍历所有进程，对于每个尚未完成的进程i，检查其所需求的资源数量Need[i][j]是否小于等于当前可用的资源数量Work[j]。如果进程i满足资源需求关系，则将其标记为完成状态，并将其分配的资源加入到可用资源数组Work中。同时将完成的进程序列号添加到正确执行的进程序列数组p中。如果存在有多个进程均满足安全性条件，并形成一个安全序列，则输出系统安全的消息，并输出正确的安全序列。如果所有进程均已经遍历完成，但仍有未满足资源需求条件的进程，则系统处于不安全状态，返回0。