**实验四 文件系统-代码阅读并调试实验**

**任务：**

**1、阅读下面源代码，写出程序功能。**

**2、阅读代码，分析代码中算法并说明。**

**3、调试并运行代码，写出结果。**

**源程序如下：**

#define \_CRT\_SECURE\_NO\_WARNINGS 1

#include<stdio.h>

#include<stdlib.h>

int Available[10]; //可使用资源向量

int Max[10][10]; //最大需求矩阵

int Allocation[10][10] = { 0 }; //分配矩阵

int Need[10][10] = { 0 }; //需求矩阵

int Work[10]; //工作向量

int Finish[10]; //是否有足够的资源分配，状态标志

int Request[10][10]; //进程申请资源向量

int Pause[10];

int arr[] = { 0 }; //各类资源总数

int List[10];

int i, j;

int n; //系统资源总数

int m; //总的进程数

int a; //当前申请的进程号

int l, e; //计数器

int b = 0, c = 0, f = 0, g; //计数器

int z = 0;

int securitycheck() //安全性检测

{

printf("\n\n");

printf("\t\t\t※ 安全性检测 ※\n\n");

if (n == 3)

{

printf(" 工作向量 尚需求量 已分配 工作向量+已分配\n进程 ");

for (c = 1; c <= 4; c++)

{

for (j = 1; j <= n; j++)

{

printf(" %d类", j);

}

}

}

if (n == 2)

{

printf(" 工作向量 尚需求量 已分配 工作向量+已分配\n进程 ");

for (c = 1; c <= 4; c++)

{

for (j = 1; j <= n; j++)

{

printf(" %d类", j);

}

}

}

for (i = 1; i <= m; i++)

{

Pause[i] = Available[i]; //Pause[i]只是一个暂时寄存的中间变量，为防止在下面安全性检查时修改到Available[i]而代替的一维数组

Finish[i] = false;

}

for (g = 1; g <= m; g++)

{

for (i = 1; i <= m; i++)

{

b = 0; //计数器初始化

Finish[i] == false;

for (j = 1; j <= n; j++)

{

if (Need[i][j] <= Pause[j])

{

b = b + 1;

}

if (Finish[i] == false && b == n)

{

Finish[i] = true;

printf("\nP[%d] ", i); //依次输出进程安全序列

for (l = 1; l <= n; l++)

{

printf(" %2d ", Pause[l]);

}

for (j = 1; j <= n; j++)

{

printf(" %2d ", Need[i][j]);

}

for (j = 1; j <= n; j++)

{

//Allocation[i][j]=Pause[j]-Need[i][j];

printf(" %2d ", Allocation[i][j]);

}

for (j = 1; j <= n; j++)

{

printf(" %2d ", Pause[j] + Allocation[i][j]);

}

for (l = 1; l <= n; l++)

{

Pause[l] = Pause[l] + Allocation[i][l];

}

}

}

}

}

printf("\n\n");

for (i = 1; i <= m; i++)

{

if (Finish[i] == true) f = f + 1; //统计Finish[i]＝＝true的个数

}

if (f == m)

{

printf("safe state");

printf("\n\n系统剩余资源量： ");

for (i = 1; i <= n; i++)

{

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

}

f = 0; //将计数器f重新初始化，为下一次提出新的进程申请做准备

return 1;

}

else

{

printf("unsafe state ");

for (i = 1; i <= n; i++)

{

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

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

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

}

return 0;

}

}

void initialize() //初始化

{

printf("请输入系统的资源种类数：");

scanf("%d", &n);

for (i = 1; i <= n; i++)

{

printf("第%d类资源总数：", i);

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

}

printf("请输入进程总数：");

scanf("%d", &m);

for (i = 1; i <= m; i++)

{

for (j = 1; j <= n; j++)

{

printf("进程P[%d]对第%d类资源的最大需求量：", i, j);

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

}

}

for (i = 1; i <= m; i++)

{

for (j = 1; j <= n; j++)

{

printf("进程P[%d]对第%d类资源已分配数：", i, j);

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

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

}

}

for (i = 1; i <= n; i++)

{

for (j = 1; j <= m; j++)

{

arr[i] -= Allocation[j][i];

}

}

for (i = 1; i <= n; i++)

Available[i] = arr[i];

securitycheck();

}

void mainrequest() //进程申请资源

{

printf("请输入申请资源的进程：");

scanf("%d", &a);

for (i = 1; i <= n; i++)

{

printf("请输入进程P[%d]对%d类资源的申请量：", a, i);

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

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

{

printf("\n出错！进程申请的资源数多于它自己申报的最大需求量\n");

return;

}

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

{

printf("\nP[%d]请求的资源数大于可用资源数，必须等待\n", a);

return;

}

}

for (i = 1; i <= n; i++)

{

//以下是试探性分配

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

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

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

}

int ret=securitycheck();

if (ret == 1)

{

int key = 0;

for (j = 1; j <= n; j++)

{

if (Need[a][j] == 0)

{

key++;

}

}

if (key == n)

{

for (j = 1; j <= n; j++)

{

Available[j] += Allocation[a][j];

Allocation[a][j] = 0;

}

}

}

}

void mainshow()

{

printf("\n\n");

if (n == 3)

{

printf(" 已分配 最大需求量 尚需要量 \n进程");

}

if (n == 2)

{

printf(" 已分配 最大需求 尚需要量 \n进程");

}

for (i = 1; i <= 3; i++)

{

for (j = 1; j <= n; j++)

{

printf(" %d类", j);

}

}

for (i = 1; i <= m; i++)

{

printf("\nP[%d]", i);

for (j = 1; j <= n; j++)

{

printf(" %2d ", Allocation[i][j]);

}

for (j = 1; j <= n; j++)

{

printf(" %2d ", Max[i][j]);

}

for (j = 1; j <= n; j++)

{

printf(" %2d ", Need[i][j]);

}

}

printf("\n\n系统剩余资源量： ");

for (i = 1; i <= n; i++)

{

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

}

printf("\n");

}

void menu()

{

printf("\n\n\t\t卐卍※§ 银行家算法 §※卐卍\n");

printf("\n\n\t\t\t1:初始化");

printf("\n \t\t\t2:进程进行资源申请");

printf("\n \t\t\t3:资源分配状态");

printf("\n \t\t\t4:退出程序");

printf("\n\n\t\t\t\t\t 请输入你的选择: ");

}

int main()

{

int key = 0;

printf("\n\n");

while (1)

{

menu();

scanf("%d", &key);

printf("\n\n");

switch (key)

{

case 1:

initialize();

break;

case 2:

mainrequest();

break;

case 3:

mainshow();

break;

case 4:

printf("\n\n\t\t\t谢谢使用 \n");

printf("\n\t\t\tSee you next time !\n\n\n");

system("pause");

return 0;

}

}

system("pause");

return 0;

}