**广州大学学生实验报告**

开课学院及实验室: 计算机科学与网络工程学院 2024年 5 月 13 日

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 学院 | 计算机科学与网络工程学院 | 年级/专业/班 |  | 姓名 |  | 学号 |  |
| 实验课程名称 | 操作系统实验 | | | | | 成绩 |  |
| 实验项目名称 | 银行家算法的实现 | | | | | 指导老师 |  |

**实验一 银行家算法的实现**

1. **实验目的**

**1、了解什么是操作系统安全状态和不安全状态；**

**2、了解如何避免系统死锁；**

**3、理解银行家算法是一种最有代表性的避免死锁的算法，掌握其实现原理及实现过程**

**二、基本要求**

1. **画出银行家算法流程图；**
2. **对算法所用的数据结构进行说明；**
3. **测试数据随机产生。不可手工输入；**
4. **编写程序并调试；**
5. **多次测试程序，截屏输出实验结果；**
6. **根据实验结果与理论课讲述的原理进行实验分析。**

**三、实验软件**

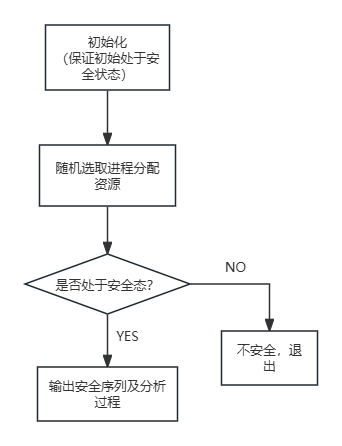
Windows操作系统、visual studio community 2022

**四、实验内容**

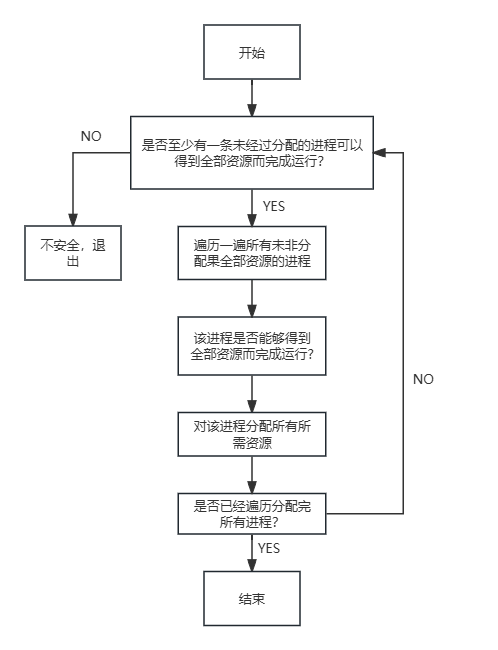
**根据银行家算法的基本思想，编写和调试一个实现动态资源分配的模拟程序，并能够有效避免死锁的发生。**

**1.算法流程图:**

**总流程图:**



**安全性判断流程图:**

****

**2.数据结构说明:**

1. **int** N = NULL;           //进程数
2. **int** M = NULL;           //资源数
3. **int**\* resource = NULL;   //资源总数
4. **int**\*\* Max = NULL;       //每个进程对应的资源需求数
5. **int**\*\* Need = NULL;      //每个进程当前需要的资源数
6. **int**\* available = NULL;  //当前所剩资源数
7. **int**\*\* allocation = NULL;//已分配资源
8. **int**\* processlist = NULL;//安全进程序列

**3.随机生成测试数据:**

1. **void** Init()
2. {
3. srand(time(NULL));
4. **bool** is\_safe = **false**;
5. N = rand() % 3 + 3;//3到5
6. M = rand() % 4 + 4;//4到7
7. //分配空间
8. resource = **new** **int**[M];
9. available = **new** **int**[M];
10. allocation = **new** **int**\* [N];
11. Max = **new** **int**\* [N];
12. Need = **new** **int**\* [N];
13. **for** (**int** i = 0; i < M; ++i)
14. {
15. allocation[i] = **new** **int** [M];
16. Max[i] = **new** **int**[M];
17. Need[i] = **new** **int**[M];
18. }
19. //随机生成数据
20. **while** (!is\_safe)
21. {
22. **for** (**int** i = 0; i < M; ++i)
23. {
24. resource[i] = rand() % 5 + 5;//5到9
25. //初始化可以资源=总资源
26. available[i] = resource[i];
27. }
28. **for** (**int** i = 0; i < N; ++i)
29. {
30. **for** (**int** j = 0; j < M; ++j)
31. {
32. **int** p = 0;
33. **int** sign1 = 1;
34. Max[i][j] = rand() % (resource[j] + 1);
35. **int** d = 0;
36. **int** sign2 = 1;
37. d = rand() % (Max[i][j] + 1);
38. allocation[i][j] = d;
39. //需求=最大需求-已经分配资源
40. Need[i][j] = Max[i][j] - allocation[i][j];
41. }
42. }
43. **for** (**int** i = 0; i < N; ++i)
44. {
45. **for** (**int** j = 0; j < M; ++j)
46. {
47. available[j] -= allocation[i][j];
48. }
49. }
50. **if** (safety(**false**))
51. {//对生成的序列进行安全检验，若安全则生成成功，否则重新生成
52. is\_safe = 1;
53. }
54. }
55. }

**解析:随机生成实验数据，但生成出来的进程序列是处于安全状态的。**

**4.核心代码:**

**对进程队列进行安全性检测**

1. **bool** safety(**bool** opt)
2. {
3. //初始化
4. **int**\* work = NULL;//当前资源所剩数量
5. **int** count = 0;//标记进程顺序
6. processlist = **new** **int**[N];//安全序列
7. **for** (**int** i = 0; i < N; ++i)processlist[i] = -1;//初始化安全序列
8. work = **new** **int**[M];
9. **for** (**int** i = 0; i < M; ++i)
10. {
11. work[i] = available[i];
12. }
13. **bool**\* visited = NULL;
14. visited = **new** **bool**[N];//判断某进程是否已经得到过全部资源，得到过为true，否则为false
15. **for** (**int** i = 0; i < N; ++i)visited[i] = **false**;
17. //输出数据=============================================================================================
18. **if** (opt)
19. {
20. cout << "---------------------------------安全检查过程---------------------------------" << endl;
21. cout << "\twork";
22. **for** (**int** i = 0; i < M \* 2-2; ++i)cout << " ";
23. cout << "Need";
24. **for** (**int** i = 0; i < M \* 2 -2; ++i)cout << " ";
25. cout << "allocation";
26. **for** (**int** i = 0; i < M \* 2 - 8; ++i)cout << " ";
27. cout << "work+allocation";
28. cout << endl;
29. }
30. //======================================================================================================
32. **for**(**int** u = 0;u<N;++u)
33. {
34. //判断是否至少有一条进程可以得到全部资源而完成运行
35. **bool**\* list = **new** **bool**[N];
36. **for** (**int** i = 0; i < N; ++i)
37. {
38. **if** (visited[i] == **false**)
39. {
40. **bool** sign = **true**;
41. **for** (**int** j = 0; j < M; ++j)
42. {
43. sign \*= (Need[i][j] <= work[j]);
44. }
45. list[i] = sign;
46. }
47. **else** list[i] = **false**;
48. }
49. **int** sign0 = 0;
50. **for** (**int** i = 0; i < N; ++i)**if** (list[i])++sign0;
51. **if** (sign0>=1)//若存在一个进程可以得到全部资源运行sing0>=1
52. {
53. //找出当前need小于work的
54. **for** (**int** i = 0; i < N; ++i)
55. {
56. **if** (!visited[i])
57. {
58. **bool** sign = **true**;
59. **for** (**int** j = 0; j < M; ++j)
60. {
61. sign \*= (Need[i][j] <= work[j]);
62. }
63. **if** (sign)//如果进程i可以全部分配
64. {
65. processlist[count] = i;
66. ++count;
67. visited[i] = **true**;
68. //输出数据=======================================================
69. **if** (opt)
70. {
71. cout << "进程" << i << "\t";
72. **for** (**int** j = 0; j < M; ++j)
73. {
74. cout << work[j] << " ";
75. }
76. cout << "  ";
77. **for** (**int** j = 0; j < M; ++j)
78. {
79. cout << Need[i][j] << " ";
80. }
81. cout << "  ";
82. **for** (**int** j = 0; j < M; ++j)
83. {
84. cout << allocation[i][j] << " ";
85. }
86. cout << "  ";
87. **for** (**int** j = 0; j < M; ++j)
88. {
89. cout << allocation[i][j] + work[j] << " ";
90. }
91. cout << endl;
92. }
93. //===============================================================
94. **for** (**int** j = 0; j < M; ++j)
95. {
96. work[j] += allocation[i][j];
97. }
98. **break**;
99. }
100. }
101. }
103. }
104. **else**
105. {//处于不安全状态
106. **return** **false**;
107. }
108. }
109. **if** (opt)cout << "------------------------------------------------------------------------------" << endl;
110. **return** **true**;
111. }

**解析:对进程序列进行安全性检查，若安全返回true，否则返回false，该函数有一个参数，bool opt，若opt为true，则该函数会打印分析过程，否则不打印分析过程。**

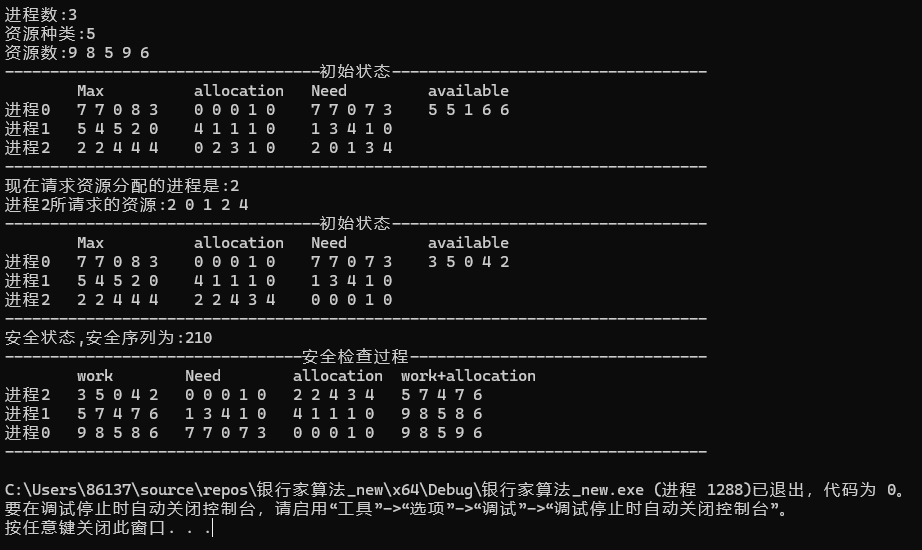
**BANK(银行家算法入口):**

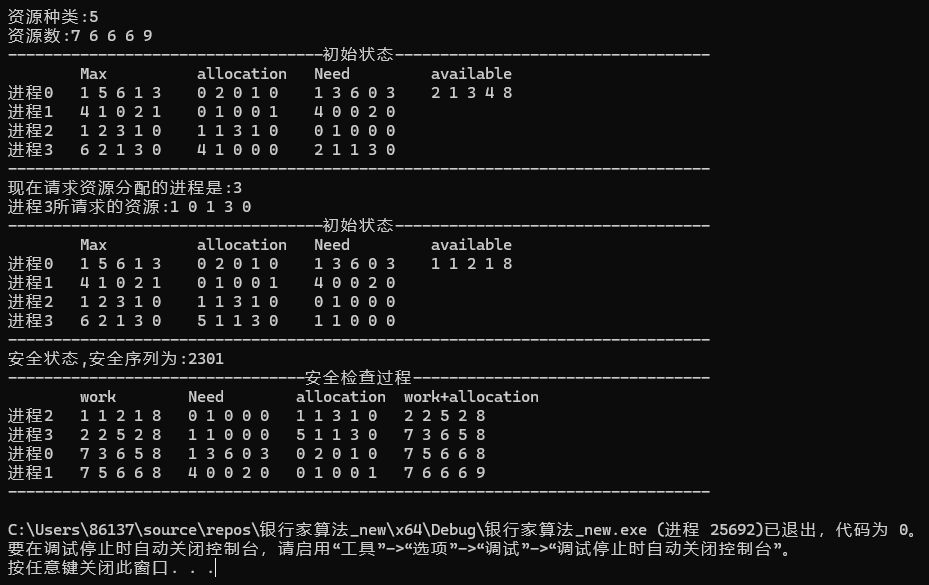
1. **void** BANK()
2. {
3. srand(time(NULL));
4. Init();
5. cout << "进程数:" << N << endl;
6. cout << "资源种类:" << M << endl;
7. cout << "资源数:";
8. **for** (**int** i = 0; i < M; ++i)cout << resource[i] << " ";
9. cout << endl;
10. print();//输出各个进程当前的资源分配状态，该函数在源码中自取
11. //随机选取一个进程分配资源
12. **int** pid = rand() % N;
13. **int**\* request = **new** **int**[M];
14. **for** (**int** i = 0; i < M; ++i)
15. {
16. request[i] = rand() % (min(Need[pid][i]+1, available[i]+1));
17. }
18. cout << "现在请求资源分配的进程是:" << pid << endl;
19. cout <<"进程"<<pid << "所请求的资源:";
20. **for** (**int** i = 0; i < M; ++i)
21. {
22. cout << request[i] << " ";
23. }
24. cout << endl;
25. **for** (**int** i = 0; i < M; ++i)
26. {
27. available[i] -= request[i];
28. Need[pid][i] -= request[i];
29. allocation[pid][i] += request[i];
30. }
31. print();//输出各个进程当前的资源分配状态
32. **bool** sign = safety(**false**);//安全性检查
33. **if** (sign)//若为安全状态
34. {
35. cout << "安全状态,安全序列为:";
36. **for** (**int** i = 0; i < N; ++i)
37. {
38. cout << processlist[i];
39. }
40. cout << endl;
41. safety(**true**);//输出检查过程
42. }
43. **else**
44. {
45. cout << "状态不安全!";//否则
46. }
47. }

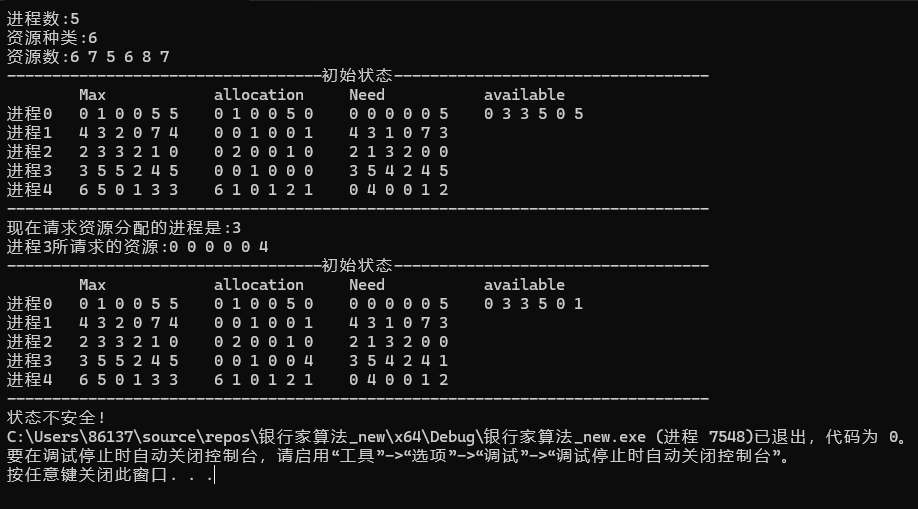
**解析:该函数，主要实现生成一个安全状态的进程队列，再随机让某进程请求资源分配(请求的分配可能是该进程所需的所有资源，也可能不是)，若此时进程处于安全状态则可以继续请求分配(该步骤未进行，若要实现增加一个循环即可)，否则进程队列不安全，退出。**

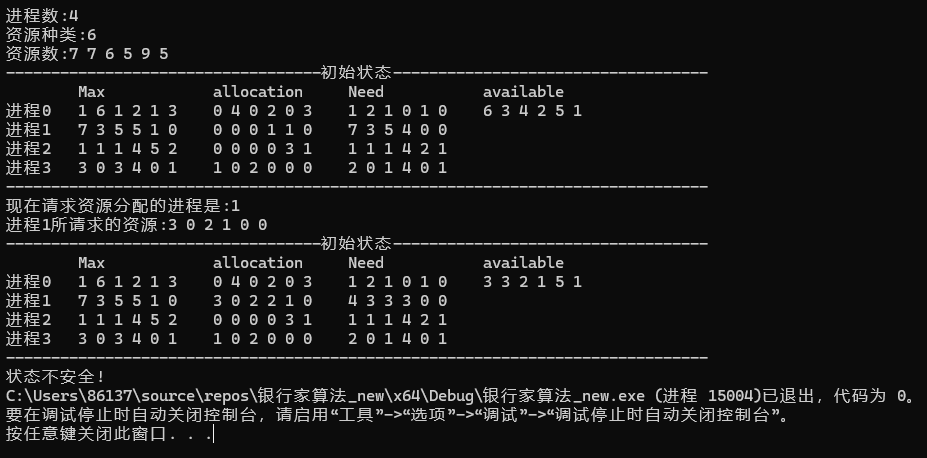
**5.实验结果演示:**

****

****

****

****

****

**五、实验思考**

**1.如何设计程序的输入模块才能满足实验要求，请举例说明；**

**答:**

**①某进程所需的最大资源，不能超过资源总量Max<=resource；**

**②初始化时已分配资源不能超过该进程所需的最大资源allocation<=Max；**

**③Need = Max – allocation；**

**④available = resource – allocation；**

**⑤该实验初始化得到的数据是处于安全状态的(但也可以是不安全状态，该实验未进行这方面探究)，该实验是生成安全队列，从中请求分配一小部分资源，再判断是否安全的一个过程。**

**2.银行家算法在实现过程中必须注意哪些资源分配细节才能避免死锁？**

**答:**

**需要比较系统每类资源数量和进程需要最大资源类，进程需要的资源量必须不超过系统可分配的资源量，如果超过了就必须退出去，系统再和下一个进程匹配，不满足请求的进程返回系统分配资源，直到全部进程匹配完毕，需要让系统得到一个安全序列，系统给每个进程分配资源达到最大资源，让每个进程都能顺利完成，来避免死锁。**