

# BankerFinally

```
1  /*可读取文件的银行家算法 */
2  #include <iostream>
3  #include <fstream>
4  #include <vector>
5
6  using namespace std;
7
8  const int MAX_PROCESS = 10; // 最大进程数
9  const int MAX_RESOURCE = 10; // 最大资源数
10
11 int n, m; // 进程数和
    资源数
12 int available[MAX_RESOURCE]; // 可用资源
    数
13 int max_need[MAX_PROCESS][MAX_RESOURCE]; // 最大需求
    量
14 int allocation[MAX_PROCESS][MAX_RESOURCE]; // 已分配资
    源量
15 int need[MAX_PROCESS][MAX_RESOURCE]; // 需求资源
    量
16 bool finish[MAX_PROCESS]; // 进程是否
    完成
17
18 vector<int> safe_seq; // 安全序列
19
20 // 从文件中读取数据并初始化银行家算法相关数据结构
21 void init(const char *filename)
22 {
23     ifstream infile(filename);
24
25     if (!infile)
26     {
27         cout << "无法打开文件 " << filename << ", 请检查文
            件路径是否正确!" << endl;
28         exit(1);
```

```
29     }
30
31     cout << "正在读取文件 " << filename << "... " <<
endl;
32
33     infile >> n >> m;
34     cout << "读取到 " << n << " 个进程和 " << m << " 种资源" << endl;
35
36     for (int i = 0; i < m; i++)
37     {
38         infile >> available[i];
39         cout << "资源" << i << " 的可用数量为 " <<
available[i] << endl;
40     }
41
42     for (int i = 0; i < n; i++)
43     {
44         cout << "读取进程" << i << "的资源需求量: " <<
endl;
45         for (int j = 0; j < m; j++)
46         {
47             infile >> max_need[i][j];
48             cout << "进程" << i << " 需要 " <<
max_need[i][j] << " 个资源" << j << endl;
49         }
50     }
51
52     for (int i = 0; i < n; i++)
53     {
54         cout << "读取进程" << i << "的已分配资源量: " <<
endl;
55         for (int j = 0; j < m; j++)
56         {
57             infile >> allocation[i][j];
58             need[i][j] = max_need[i][j] -
allocation[i][j];
59             if (need[i][j] < 0)
60             {
```

```

61         cout << "错误：已分配的资源量大于最大需求
量！" << endl;
62         exit(1);
63     }
64     cout << "进程" << i << " 已分配 " <<
allocation[i][j] << " 个资源" << j << ", 还需要 " <<
need[i][j] << " 个" << endl;
65     }
66     finish[i] = false;
67 }
68
69 infile.close();
70
71 cout << "文件读取成功！" << endl;
72 }
73
74 // 判断是否满足需求
75 bool check_need(int process_id)
76 {
77     for (int i = 0; i < m; i++)
78     {
79         if (need[process_id][i] > available[i])
80         {
81             return false;
82         }
83     }
84     return true;
85 }
86
87 // 模拟分配资源
88 void simulate()
89 {
90     int count = 0; // 已完成进程数
91     while (count < n)
92     {
93         bool found = false;
94         for (int i = 0; i < n; i++)
95         {
96             if (!finish[i] && check_need(i))
97                 { // 满足需求

```

```

198         for (int j = 0; j < m; j++)
199         {
200             available[j] += allocation[i][j];
201             // 释放资源
202         }
203         finish[i] = true;
204         count++;
205         safe_seq.push_back(i); // 将进程加入安全
206         // 序列
207         found = true;
208     }
209 }
210 if (!found)
211 { // 没有进程满足需求, 说明不安全
212     cout << "错误: 系统不安全! 无法找到安全序列" <<
213     endl;
214     exit(1);
215 }
216 }
217 }
218 }
219
220 void print_result()
221 {
222     cout << "找到安全序列, 进程可以成功执行!" << endl;
223     cout << "安全序列为: ";
224     for (int i = 0; i < n; i++)
225     {
226         cout << "P" << safe_seq[i];
227         if (i != n - 1)
228         {
229             cout << " -> ";
230         }
231     }
232     cout << endl;
233
234     cout << "进程    最大需求量    已分配资源量    ";
235     for (int i = 0; i < m; i++)
236     {
237         cout << "还需资源量" << i << "    ";
238     }

```

```

135     cout << endl;
136
137     for (int i = 0; i < n; i++)
138     {
139         cout << " P" << i << " ";
140         for (int j = 0; j < m; j++)
141         {
142             cout << max_need[i][j] << "-----"
143             << allocation[i][j] << "-----" << need[i][j]
144             << " ";
145         }
146         cout << endl;
147     }
148     cout << endl;
149
150     cout << "系统分配资源的过程如下: " << endl;
151
152     for (int i = 0; i < n; i++)
153     {
154         cout << "系统分配资源给进程 P" << safe_seq[i] <<
155         ":" << endl;
156         for (int j = 0; j < m; j++)
157         {
158             cout << " 将 " << allocation[safe_seq[i]]
159             [j] << " 个资源" << j << " 分配给进程 P" << safe_seq[i]
160             << endl;
161             cout << " 进程 P" << safe_seq[i] << " 现在
162             已经得到 " << allocation[safe_seq[i]][j] << " 个资源" <<
163             j << endl;
164         }
165         cout << " 因此, 进程 P" << safe_seq[i] << " 现
166         在的资源情况为: " << endl;
167         cout << " ";
168         for (int j = 0; j < m; j++)
169         {
170             cout << "资源" << j << " : " <<
171             allocation[safe_seq[i]][j] << " ";
172         }
173         cout << endl;
174     }
175     << endl;

```

```

166     }
167 }
168
169 int main()
170 {
171     const char *filename = "data.txt";
172     init(filename);
173     simulate();
174     print_result();
175     return 0;
176 }

```

进程	最大需求量	已分配资源量	还需资源量0	还需资源量1	还需资源量2
P0	7	0	5	1	4
P1	3	2	2	0	2
P2	9	3	0	0	0
P3	2	2	2	1	1

系统分配资源的过程如下:

系统分配资源给进程 P1:

将 2 个资源0 分配给进程 P1  
 进程 P1 现在已经得到 2 个资源0  
 将 0 个资源1 分配给进程 P1  
 进程 P1 现在已经得到 0 个资源1  
 将 0 个资源2 分配给进程 P1  
 进程 P1 现在已经得到 0 个资源2  
 因此, 进程 P1 现在的资源情况为:  
 资源0 : 2 资源1 : 0 资源2 : 0

资源0

资源1

资源2

系统分配资源给进程 P3:

将 2 个资源0 分配给进程 P3  
 进程 P3 现在已经得到 2 个资源0  
 将 1 个资源1 分配给进程 P3  
 进程 P3 现在已经得到 1 个资源1  
 将 1 个资源2 分配给进程 P3  
 进程 P3 现在已经得到 1 个资源2  
 因此, 进程 P3 现在的资源情况为:  
 资源0 : 2 资源1 : 1 资源2 : 1

系统分配资源给进程 P0:

将 0 个资源0 分配给进程 P0  
 进程 P0 现在已经得到 0 个资源0  
 将 1 个资源1 分配给进程 P0  
 进程 P0 现在已经得到 1 个资源1  
 将 0 个资源2 分配给进程 P0  
 进程 P0 现在已经得到 0 个资源2  
 因此, 进程 P0 现在的资源情况为:  
 资源0 : 0 资源1 : 1 资源2 : 0

系统分配资源给进程 P2:

将 3 个资源0 分配给进程 P2  
 进程 P2 现在已经得到 3 个资源0  
 将 0 个资源1 分配给进程 P2  
 进程 P2 现在已经得到 0 个资源1  
 将 2 个资源2 分配给进程 P2  
 进程 P2 现在已经得到 2 个资源2  
 因此, 进程 P2 现在的资源情况为:  
 资源0 : 3 资源1 : 0 资源2 : 2

CorrectBanker.cpp

data.txt

×

newBanker.cpp

OSLab\_Lhh > Lab4\_Banker > data.txt

4个进程, 3种资源

1 4 3

2 3 3 2

当前资源剩余情

3 7 5 3

4 3 2 2

最大资源需求矩阵

5 9 0 2

6 2 2 2

7 0 1 0

8 2 0 0

9 3 0 2

已分配资源矩阵

10 2 1 1