

Project6 实验报告

余北辰 519030910245

1 实验概述

1.1 实验名称

Banker's Algorithm

1.2 实验内容

1. 实现银行家算法
2. 维护available, maximum, allocation, need四个资源数组，通过 * 命令打印各数组的值
3. 通过 RQ 命令请求资源、RL 命令释放资源

2 实验环境

- Ubuntu 18.04.5 LTS
- Linux version 5.4.0-72-generic
- VirtualBox 6.1.18

3 实验过程与结果展示

3.1 安全状态检验

银行家算法需要四个资源数组：

```
1  int available[NUMBER_OF_RESOURCES];
2  int maximum[NUMBER_OF_CUSTOMERS][NUMBER_OF_RESOURCES];
3  int allocation[NUMBER_OF_CUSTOMERS][NUMBER_OF_RESOURCES];
4  int need[NUMBER_OF_CUSTOMERS][NUMBER_OF_RESOURCES];
```

available 数组存储各种资源剩余可用的数量；

maximum 数组存储每个客户对每种资源的总需求;

allocation 数组存储每个客户当前分配到的每种资源的数量;

need 数组存储每个客户对每种资源的剩余需求。

安全状态检验函数 **check_safe()** 需要引入 **finish** 与 **work** 两个数组。 **finish** 数组标记各个客户是否已经完成对资源的使用; **work** 数组则储存各资源的当前可用数量;

首先让 **finish** 数组初始化为0, **work** 数组则初始化为 **available** 数组的值。

循环检查各客户的剩余需求能否被当前空闲资源所满足; 若能够满足, 则先将空闲资源分配予其, 待其使用完毕后释放其现有资源。由于我们只是模拟银行家算法, 因此省略中间过程, 直接将其现有资源释放, 即 **work** 数组加上该客户的 **allocation** 数组。该客户的 **finish** 数组被标记为1。

不断循环上面的过程, 直到所有的客户的 **finish** 数组都被标记为1, **check_safe()** 返回0, 或者当前空闲资源无法满足剩下任何一个客户的需求了, **check_safe()** 返回-1。

代码具体实现如下:

```
1  int check_safe(int allocation[][NUMBER_OF_RESOURCES], int need[]  
   [NUMBER_OF_RESOURCES], int available[])  
2  {  
3      int safe = 0;  
4      int finish[NUMBER_OF_CUSTOMERS];  
5      int work[NUMBER_OF_RESOURCES];  
6      for (int i = 0; i < NUMBER_OF_CUSTOMERS; ++i)  
7      {  
8          finish[i] = 0;  
9      }  
10     for (int i = 0; i < NUMBER_OF_RESOURCES; ++i)  
11     {  
12         work[i] = available[i];  
13     }  
14     int all_finished;  
15     int all_checked;  
16     while (1)  
17     {  
18         all_finished = 1;  
19         all_checked = 1;  
20         for (int i = 0; i < NUMBER_OF_CUSTOMERS; ++i)  
21         {  
22             if (finish[i] == 0)  
23                 all_finished = 0;  
24             else
```

```

25         continue;
26
27     int flag = 1;
28     for (int j = 0; j < NUMBER_OF_RESOURCES; ++j)
29     {
30         if (need[i][j] > work[j])
31             flag = 0;
32     }
33
34     if (flag)
35     {
36         all_checked = 0;
37         for (int j = 0; j < NUMBER_OF_RESOURCES; ++j)
38         {
39             work[j] += allocation[i][j];
40             finish[i] = 1;
41         }
42     }
43 }
44
45 if (all_checked || all_finished)
46     break;
47 }
48 if (all_finished == 1)
49     return 0;
50 else
51     return -1;
52 }

```

3.2 资源请求

在请求资源时，先要调用安全状态检验函数，保证处于安全状态才分配资源；否则不予分配。

```

1  int request_resources(int num, int request[])
2  {
3      if (num < 0 || num >= NUMBER_OF_CUSTOMERS)
4      {
5          fprintf(stderr, "Error customer id!\n");
6          return -1;
7      }
8      int available_[NUMBER_OF_RESOURCES];
9      int allocation_[NUMBER_OF_CUSTOMERS][NUMBER_OF_RESOURCES];
10     int need_[NUMBER_OF_CUSTOMERS][NUMBER_OF_RESOURCES];

```

```

11
12     for (int i = 0; i < NUMBER_OF_RESOURCES; ++i)
13     {
14         available_[i] = available[i];
15         for (int j = 0; j < NUMBER_OF_CUSTOMERS; ++j)
16         {
17             allocation_[j][i] = allocation[j][i];
18             need_[j][i] = need[j][i];
19         }
20     }
21
22     for (int i = 0; i < NUMBER_OF_RESOURCES; ++i)
23     {
24         allocation_[num][i] += request[i];
25         need_[num][i] -= request[i];
26         available_[i] -= request[i];
27     }
28
29     int safe = check_safe(allocation_, need_, available_);
30     if (safe == 0)
31     {
32         for (int i = 0; i < NUMBER_OF_RESOURCES; ++i)
33         {
34             allocation[num][i] += request[i];
35             need[num][i] -= request[i];
36             available[i] -= request[i];
37         }
38         printf("Safe. Resource allocation succeeded.\n");
39         return 0;
40     }
41     else
42     {
43         fprintf(stderr, "Unsafe! Resource allocation failed!\n");
44         return -1;
45     }

```

3.3 资源释放

资源释放的过程相比之下简单一些，只需要检查一下是否有足够的资源供给释放，再释放所选资源即可。

```

1     int release_resources(int num, int release[])
2     {
3         if (num < 0 || num >= NUMBER_OF_CUSTOMERS)

```

```

4      {
5          fprintf(stderr, "Error customer id!\n");
6          return -1;
7      }
8      for (int i = 0; i < NUMBER_OF_RESOURCES; ++i)
9      {
10         if (allocation[num][i] < release[i])
11         {
12             fprintf(stderr, "Don't have so much to release!\n");
13             return -1;
14         }
15     }
16     for (int i = 0; i < NUMBER_OF_RESOURCES; ++i)
17     {
18         available[i] += release[i];
19         need[num][i] += release[i];
20         allocation[num][i] -= release[i];
21     }
22     return 0;
23 }

```

3.4 其他部分

打印资源数组:

```

1      void print_resources()
2      {
3          printf("Available:\n");
4          printf(" \tA\tB\tC\tD\n");
5          printf(" \t");
6          for (int i = 0; i < NUMBER_OF_RESOURCES; ++i)
7          {
8              printf("%d\t", available[i]);
9          }
10         printf("\n");
11         printf("\n");
12         printf("Maximum:\n");
13         printf(" \tA\tB\tC\tD\n");
14         for (int i = 0; i < NUMBER_OF_CUSTOMERS; ++i)
15         {
16             printf("P%d\t", i);
17             for (int j = 0; j < NUMBER_OF_RESOURCES; ++j)
18             {
19                 printf("%d\t", maximum[i][j]);

```

```

20     }
21     printf("\n");
22 }
23 printf("\n");
24 printf("\n");
25 printf("Allocation:\n");
26 printf(" \tA\tB\tC\tD\n");
27 for (int i = 0; i < NUMBER_OF_CUSTOMERS; ++i)
28 {
29     printf("P%d\t", i);
30     for (int j = 0; j < NUMBER_OF_RESOURCES; ++j)
31     {
32         printf("%d\t", allocation[i][j]);
33     }
34     printf("\n");
35 }
36 printf("\n");
37 printf("\n");
38 printf("Need:\n");
39 printf(" \tA\tB\tC\tD\n");
40 for (int i = 0; i < NUMBER_OF_CUSTOMERS; ++i)
41 {
42     printf("P%d\t", i);
43     for (int j = 0; j < NUMBER_OF_RESOURCES; ++j)
44     {
45         printf("%d\t", need[i][j]);
46     }
47     printf("\n");
48 }
49 printf("\n");
50 }

```

input.txt 文件的读取，通过 **read_file()** 函数获得各资源数组的初始值：

```

1  int read_file()
2  {
3      FILE *f;
4      f = fopen("input.txt", "r");
5      if (f == NULL)
6      {
7          fprintf(stderr, "File not found!\n");
8          return 0;
9      }
10     else

```

```

11     {
12         for (int i = 0; i < NUMBER_OF_CUSTOMERS; ++i)
13             for (int j = 0; j < NUMBER_OF_RESOURCES; ++j)
14                 {
15                     fscanf(f, "%d", &maximum[i][j]);
16                     fgetc(f);
17                 }
18         for (int i = 0; i < NUMBER_OF_CUSTOMERS; ++i)
19             for (int j = 0; j < NUMBER_OF_RESOURCES; ++j)
20                 {
21                     need[i][j] = maximum[i][j];
22                     allocation[i][j] = 0;
23                 }
24         return 0;
25     }
26 }

```

main() 函数中，需要完成对用户的命令的识别与选择：

```

1  int main(int argc, char *argv[])
2  {
3      if (argc != 5)
4      {
5          fprintf(stderr, "Error input!\n");
6          return -1;
7      }
8
9      for (int i = 1; i <= NUMBER_OF_RESOURCES; ++i)
10     {
11         available[i - 1] = atoi(argv[i]);
12     }
13
14     if (read_file() != 0)
15         return -1;
16
17     while (1)
18     {
19         printf(">>>");
20         char ch[10];
21         scanf("%s", ch);
22         if (strcmp(ch, "exit") == 0)
23         {
24             break;
25         }

```

```

26     else if (strcmp(ch, "*") == 0)
27     {
28         print_resources();
29         continue;
30     }
31     else if (strcmp(ch, "RQ") == 0)
32     {
33         int num;
34         scanf("%d", &num);
35         int request[NUMBER_OF_RESOURCES];
36         for (int i = 0; i < NUMBER_OF_RESOURCES; ++i)
37         {
38             scanf("%d", &request[i]);
39         }
40         request_resources(num, request);
41     }
42     else if (strcmp(ch, "RL") == 0)
43     {
44         int num;
45         scanf("%d", &num);
46         int release[NUMBER_OF_RESOURCES];
47         for (int i = 0; i < NUMBER_OF_RESOURCES; ++i)
48         {
49             scanf("%d", &release[i]);
50         }
51         release_resources(num, release);
52     }
53     else
54     {
55         fprintf(stderr, "Error input!\n");
56     }
57 }
58
59 return 0;
60 }

```

3.5 测试结果

`input.txt` 如下：

- 1 6,4,7,3
- 2 4,2,3,2
- 3 2,5,3,3
- 4 6,3,3,2
- 5 5,6,7,5

测试资源请求：

```
polaris@polaris-VirtualBox: ~/course/Operating-Systems/Project/Project6
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
polaris@polaris-VirtualBox:~/course/Operating-Systems/Project/Project6$ ./banker 9 9 9 9
>>>RQ 0 5 2 4 3
Safe. Resource allocation succeeded.
>>>*
Available:
      A      B      C      D
      4      7      5      6

Maximum:
      A      B      C      D
P0     6      4      7      3
P1     4      2      3      2
P2     2      5      3      3
P3     6      3      3      2
P4     5      6      7      5

Allocation:
      A      B      C      D
P0     5      2      4      3
P1     0      0      0      0
P2     0      0      0      0
P3     0      0      0      0
P4     0      0      0      0

Need:
      A      B      C      D
P0     1      2      3      0
P1     4      2      3      2
P2     2      5      3      3
P3     6      3      3      2
P4     5      6      7      5

>>>|
```

测试资源释放：

```
polaris@polaris-VirtualBox: ~/course/Operating-Systems/Project/Project6
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
polaris@polaris-VirtualBox:~/course/Operating-Systems/Project/Project6$ ./banker 9 9 9 9
>>>RQ 0 5 2 4 3
Safe. Resource allocation succeeded.
>>>RL 0 1 1 1 1
>>>*
Available:
      A      B      C      D
      5      8      6      7

Maximum:
      A      B      C      D
P0     6      4      7      3
P1     4      2      3      2
P2     2      5      3      3
P3     6      3      3      2
P4     5      6      7      5

Allocation:
      A      B      C      D
P0     4      1      3      2
P1     0      0      0      0
P2     0      0      0      0
P3     0      0      0      0
P4     0      0      0      0

Need:
      A      B      C      D
P0     2      3      4      1
P1     4      2      3      2
P2     2      5      3      3
P3     6      3      3      2
P4     5      6      7      5

>>>|
```

测试不安全状态的检验：

```
polaris@polaris-VirtualBox: ~/course/Operating-Systems/Project/Project6
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
polaris@polaris-VirtualBox:~/course/Operating-Systems/Project/Project6$ ./banker 5 5 5 5
>>>RQ 0 5 2 4 3
Unsafe! Resource allocation failed!
>>>
```

4 实验总结

1. 调用安全状态检验函数时，注意要将当前的资源数组拷贝一份，并使用备份的资源数组进行检验。因为安全状态检验函数是指针传递，会改变资源数组的值。

5 实验参考资料

- 实验参考书籍：Operating System Concept, 10th edition
- 实验源代码网址：<https://github.com/greggagne/osc10e>