计算机系统体系结构 Project6

517030910326 王孝诚

2019.11.30

实验环境

Windows 10 下使用 VMWare Workstation 15 Player 创建和运行虚拟机，虚拟机环境是 Linux 发行版 Ubuntu16.04.6 LTS。

1 银行家算法

用 c 语言实现了银行家算法程序。

1.1 设计思路

1. **初始化各个数据结构：** 规定 5 个用户和 4 个资源，每个资源有多少实例由用户输入，用于初始化 available 数组。max 数据由文件读入，需要处理文件输入的格式转化，然后初始化 maximum 数组。allocation 数组初始化为 0，need 数组根据 maximum 和 allocation 初始化获得。

2. **request_resources：** 每次获得操作的用户以及申请的实例数量，然后用 safety 算法检查该输入是否保证了安全性，从而决定接收或者拒绝该申请。

3. **release_resources：** 将指定的用户的资源释放指定数目，这里如果输入指定数目超过当前申请的数量，不报错，释放掉所有即可。

1.2 核心代码解释

1. 初始化

```
for(int i = 0; i < argc - 1; i++){
    available[i] = atoi(argv[i + 1]);
}
for(int i = 0; i < NUMBER_OF_CUSTOMERS; i++){
    for(int j = 0; j < NUMBER_OF_RESOURCES; j++){
        allocation[i][j] = 0;
    }
}
load_max(); // 从文件加载maximum
update_need(); // 根据maximum和allocation初始化need
```

2. request resources

```
int request_resources(int customer_num, int request[])
{
    for(int i = 0; i < NUMBER_OF_RESOURCES; i++){
```

```
4           if(request[i] > maximum[customer_num][i] -
                allocation[customer_num][i]){
5               printf("Request exceed Max\n");
6               return 0;
7           }
8       }
9       for(int i = 0; i < NUMBER_OF_RESOURCES; i++){
10          allocation[customer_num][i] += request[i];
11          available[i] -= request[i];
12      }
13      update_need();
14
15      // safety algorithm
16      int count = 0, work[NUMBER_OF_RESOURCES];
17      for(int i = 0; i < NUMBER_OF_CUSTOMERS; i++){
18          finish[i] = 0;
19      }
20      for(int i = 0; i < NUMBER_OF_RESOURCES; i++){
21          work[i] = available[i];
22      }
23
24      for(int i = 0; i < NUMBER_OF_CUSTOMERS; i++){
25          if(!finish[i]){
26              int j;
27              for(j = 0; j < NUMBER_OF_RESOURCES; j++){
28                  if(work[j] < need[i][j])
29                      break;
30              }
31              if(j == NUMBER_OF_RESOURCES){
32                  finish[i] = 1;
33                  count++;
34                  for(j = 0; j < NUMBER_OF_RESOURCES; j++)
35                      work[j] += allocation[i][j];
36              }
37          }
38      }
39      if(count == NUMBER_OF_CUSTOMERS){ // safe
40          return 0;
41      }
42      else{  // not safe
43          for(int i = 0; i < NUMBER_OF_RESOURCES; i++){
44              allocation[customer_num][i] -= request[i];
45              available[i] += request[i];
46          }
47
48          return -1;
49      }
50  }
```

3. release resources

```
1   void release_resources(int customer_num, int release[])
```

```
{
    for(int i = 0; i < NUMBER_OF_RESOURCES; i++){
        if(release[i] > allocation[customer_num][i]){
            allocation[customer_num][i] = 0;
            available[i] += allocation[customer_num][i];
        }
        else{
            allocation[customer_num][i] -= release[i];
            available[i] += release[i];
        }
    }
    update_need();
}
```

1.3 实验结果

```
xcwang@ubuntu:~/Documents/project6$ ./theBanker 10 5 7 8
cmd>RQ 0 3 1 2 1
Denied.
cmd>RL 4 1 2 3 1
cmd>*
Avaliable
10      5       7       8
Maximum
T0      6       4       7       3
T1      4       2       3       2
T2      2       5       3       3
T3      6       3       3       2
T4      5       6       7       5
Allocation
T0      0       0       0       0
T1      0       0       0       0
T2      0       0       0       0
T3      0       0       0       0
T4      0       0       0       0
Need
T0      6       4       7       3
T1      4       2       3       2
T2      2       5       3       3
T3      6       3       3       2
T4      5       6       7       5
```

Figure 1: 示例