上海交通大學

《操作系统》课程

学生实验报告

实验名称: Project6:Banker’s Algorithm

姓　　名：洪瑄锐

学　　号：517030910227

班　　级：F1703302

手　　机：15248246044

任课老师：吴晨涛

2019 年 6 月 5 日

目录

一、实验要求

编写程序实现课本 8.6.3 节中讨论的银行家算法。客户从银行请求并释放资源。银行家只有在系统处于安全状态时才会发出请求。该算法将拒绝使系统处于不安全状态的请求。

二、程序设计思想及代码解释

1. 基本数据结构及初始化

- int *available 用于存储系统各个资源的可用数量
- int **maximum 用于存储各个消费者对于每种资源的最大需求数量
- int **allocation 用于存储各个消费者已被分配的每种资源的数量
- int **need 用于存储各个消费者对于每种资源目前需要的数量

初始化：

- 分配动态空间，因课本所给示例 customer 的数量为 5，resource 的种类为 4，因此利用 malloc 函数分配空间。此外，对于 maximum 和 need 函数，根据初始化文件 init.txt 进行初始化，即读文件操作。

```
void init_array() {
    char *buffer;
    int i, j;
    int idx;
    FILE *fp = fopen(INPUT_FILEPATH, "r");
    assert(fp);

    buffer = malloc(sizeof(char) * MAX_LINE_LENGTH);

    num_customers = 5;
    num_resources = 4;

    maximum = malloc(sizeof(int *) * num_customers);
    allocation = malloc(sizeof(int *) * num_customers);
    need = malloc(sizeof(int *) * num_customers);
    available = malloc(sizeof(int) * num_resources);

    for (i = 0; i < num_customers; i++) {
        maximum[i] = malloc(sizeof(int) * num_resources);
        allocation[i] = malloc(sizeof(int) * num_resources);
        need[i] = malloc(sizeof(int) * num_resources);
        memset(allocation[i], 0, num_resources);
    }
    memset(available, 0, num_customers);

    i = j = 0;
    while (fgets(buffer, MAX_LINE_LENGTH, fp) != NULL) {
        for (idx = 0; idx < MAX_LINE_LENGTH; idx++) {
            if (buffer[idx] == ',' || buffer[idx] == '\n') {
                need[i][j] = atoi((char *)&buffer[idx - 1]);
                maximum[i][j] = need[i][j];
                j++;
            }
            if (buffer[idx] == '\n') break;
        }
        i++;
        j = 0;
    }

    fclose(fp);
    printf("Initialization finished.\n");
}
```

2. is_smaller_equal(int *a, int *b, int size)

用于比较两个一维数组的大小，如果 a 的每一个元素都比 b 对应的元素小或者二者相等，则返回 1，否则返回 0.

```
/*比较数组大小,a<=b返回1, a>b返回0*/
int is_smaller_equal(int *a, int *b, int size)
{
    int flag = 1;
    for (int i = 0; i < size; i++)
    {
        if (*(a + i) <= *(b + i));
        else
            flag = 0;
    }
    return flag;
}
```

3. is_safe()

用于判断当前状态是否安全。利用书中 8.6.3 节所给算法。

1. Let Work and Finish be vectors of length m and n, respectively. Initialize Work = Available and Finish[i] = false for i = 0, 1, ..., n − 1.
2. Find an index i such that both a. Finish[i] == false b. Needi ≤ Work If no such i exists, go to step 4.
3. Work = Work + Allocationi Finish[i] = true Go to step 2.
4. If Finish[i] == true for all i, then the system is in a safe state.

此外，对于本题目，判断安全状态函数用于测试请求是否可以满足，在判断安全状态之前已经假设该请求可满足，将请求的资源分配给了这位消费者，因此对于 allocation 均为 0 的消费者可认为它已经被完成，分配给它的资源已经被释放。

```
/*判断是否为安全状态,安全返回1, 不安全返回0*/
int is_safe()
{
    int flag[NUMBER_OF_CUSTOMERS];//标记allocation_i是否全为0,如果为0则置1
    int work[NUMBER_OF_RESOURCES];
    int finished[NUMBER_OF_CUSTOMERS];

    for (int i = 0; i < NUMBER_OF_RESOURCES; i++)
    {
        work[i] = available[i];
    }

    for (int i = 0; i < NUMBER_OF_CUSTOMERS; i++)
    {
        flag[i] = 1;
        for (int j = 0; j < NUMBER_OF_RESOURCES; j++)
        {
            if (allocation[i][j] == 0);
            else flag[i] = 0;
        }
        if (flag[i])
            finished[i] = 1;
        else
            finished[i] = 0;
    }
```

```
    while (1)
    {
        int i;
        for (i = 0; i < NUMBER_OF_CUSTOMERS; i++)
        {
            if ((finished[i] == 0) && is_smaller_equal(need[i], work, NUMBER_OF_RESOURCES))
                break;
        }
        if (i == NUMBER_OF_CUSTOMERS)
            break;
        else
        {
            for (int j = 0; j < NUMBER_OF_RESOURCES; j++)
            {
                work[j] += allocation[i][j];
                finished[i] = 1;
            }
        }
    }

    for (int j = 0; j < NUMBER_OF_CUSTOMERS; j++)
    {
        if (finished[j] == 0)
            return 0;
    }
    return 1;
}
```

4. request_resources(int customer_num,int request[])

首先判断 request 是否超过了 availab，如果超过了则拒绝这个请求，否则假设可以满足，将要求的资源分配给这位消费者，调用 is_safe()函数，判断当前状态是否安全，如果安全则可以满足，如果不安全则拒绝这个请求，并将刚才分配出去的资源收回，恢复到未请求之前的状态。

```
int request_resources(int customer_num, int request[])
{
    if (!(is_smaller_equal(request, need[customer_num],NUMBER_OF_RESOURCES)))
        return -1;
    else
    {
        if (!(is_smaller_equal(request, available, NUMBER_OF_RESOURCES)))
            return -1;
        else
        {
            for (int i = 0; i < NUMBER_OF_RESOURCES; i++)
            {
                available[i] -= request[i];
                allocation[customer_num][i] += request[i];
                need[customer_num][i] -= request[i];
            }
            if (is_safe()) return 0;
            else
            {
                for (int i = 0; i < NUMBER_OF_RESOURCES; i++)
                {
                    available[i] += request[i];
                    allocation[customer_num][i] -= request[i];
                    need[customer_num][i] += request[i];
                }
                return -1;
            }
        }
    }
}
```

5.release_resources(int customer_num,int release[])
根据参数修改 need，available,allocation 函数即可。

```
void release_resources(int customer_num, int release[])
{
    for (int i = 0; i < NUMBER_OF_RESOURCES; i++)
    {
        need[customer_num][i] += release[i];
        allocation[customer_num][i] -= release[i];
        available[i] += release[i];
    }
}
```

6.主函数
首先要获取命令行参数，初始化 available 数组。
然后获取命令，RQ 则调用 request_resources 函数，RL 则调用 release_resourses 函数，*则输出各个数组状态，exit 则结束程序.

```
int main(int argc,char *argv[])
{
    init_array();
    available[0] = atoi(argv[1]);
    available[1] = atoi(argv[2]);
    available[2] = atoi(argv[3]);
    available[3] = atoi(argv[4]);


    int request[NUMBER_OF_RESOURCES];
    int release[NUMBER_OF_RESOURCES];
    int customer_num;
    char command[20];

    while (1)
    {
        scanf("%s", command);
        if (strcmp(command, "RQ") == 0)
        {
            scanf("%d%d%d%d%d", &customer_num, &request[0], &request[1], &request[2], &request[3]);
            if (request_resources(customer_num, request))
            {
                printf("The request is denied.\n");
            }
            else
                printf("The request is satisfied.\n");
        }
        else if (strcmp(command, "RL") == 0)
        {
            scanf("%d%d%d%d%d", &customer_num, &release[0], &release[1], &release[2], &release[3]);
            release_resources(customer_num, release);
            printf("Release.\n");
        }
```

```
    else if (strcmp(command, "*") == 0)
    {
        printf("avalable:\n");
        for (int i = 0; i < NUMBER_OF_RESOURCES; i++)
        {
            printf("%d ", available[i]);
        }
        printf("\n");

        printf("maximum:\n");
        for (int i = 0; i < NUMBER_OF_CUSTOMERS; i++)
        {
            printf("customer_%d: ",i);
            for (int j = 0; j < NUMBER_OF_RESOURCES; j++)
                printf("%d ", maximum[i][j]);
            printf("\n");
        }

        printf("allocation:\n");
        for (int i = 0; i < NUMBER_OF_CUSTOMERS; i++)
        {
            printf("customer_%d: ",i);
            for (int j = 0; j < NUMBER_OF_RESOURCES; j++)
                printf("%d ", allocation[i][j]);
            printf("\n");
        }

        printf("need:\n");
        for (int i = 0; i < NUMBER_OF_CUSTOMERS; i++)
        {
            printf("customer_%d: ",i);
            for (int j = 0; j < NUMBER_OF_RESOURCES; j++)
                printf("%d ", need[i][j]);
            printf("\n");
        }


     else if (strcmp(command, "exit") == 0)
         break;
  }


  return 0;
}
```

三、运行结果截图

1.初始化

```
osc@ubuntu:~/final-src-osc10e/ch8$ ./banker 10 5 7 8
Initialization finished.
*
avalable:
10 5 7 8
maximum:
customer_0: 6 4 7 3
customer_1: 4 2 3 2
customer_2: 2 5 3 3
customer_3: 6 3 3 2
customer_4: 5 6 7 5
allocation:
customer_0: 0 0 0 0
customer_1: 0 0 0 0
customer_2: 0 0 0 0
customer_3: 0 0 0 0
customer_4: 0 0 0 0
need:
customer_0: 6 4 7 3
customer_1: 4 2 3 2
customer_2: 2 5 3 3
customer_3: 6 3 3 2
customer_4: 5 6 7 5
```

2.RQ RL

```
RQ 0 3 0 0 0
The request is satisfied.
RQ 2 3 4 7 3
The request is denied.
RQ 4 1 0 0 0
The request is denied.
RQ 3 2 0 0 0
The request is satisfied.
*
avalable:
5 5 7 8
maximum:
customer_0: 6 4 7 3
customer_1: 4 2 3 2
customer_2: 2 5 3 3
customer_3: 6 3 3 2
customer_4: 5 6 7 5
allocation:
customer_0: 3 0 0 0
customer_1: 0 0 0 0
customer_2: 0 0 0 0
customer_3: 2 0 0 0
customer_4: 0 0 0 0
need:
customer_0: 3 4 7 3
customer_1: 4 2 3 2
customer_2: 2 5 3 3
customer_3: 4 3 3 2
customer_4: 5 6 7 5
```