操作系统(D)

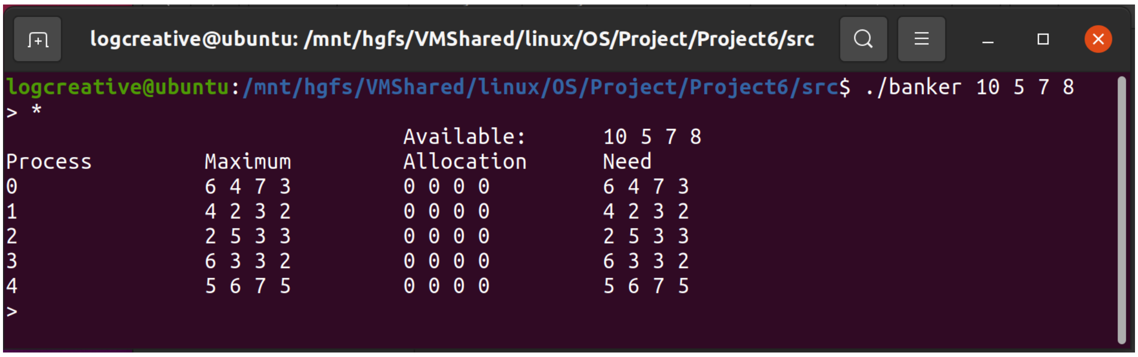
项目 6

李子龙 518070910095

2021 年 4 月 1 日

银行家算法

一 输入框架


logcreative@ubuntu: /mnt/hgfs/VMShared/linux/OS/Project/Project6/src

```
logcreative@ubuntu:/mnt/hgfs/VMShared/linux/OS/Project/Project6/src$ ./banker 10 5 7 8
> *
                                Available:      10 5 7 8
Process         Maximum         Allocation      Need
0               6 4 7 3         0 0 0 0         6 4 7 3
1               4 2 3 2         0 0 0 0         4 2 3 2
2               2 5 3 3         0 0 0 0         2 5 3 3
3               6 3 3 2         0 0 0 0         6 3 3 2
4               5 6 7 5         0 0 0 0         5 6 7 5
>
```

首先需要读取可用资源和最大使用资源数。使用了 UNIX 系统中 strsep() 函数按照 , 分割输入行。初始时：

$$\mathsf{maximum}_i = \mathsf{need}_i$$

$$\mathsf{allocation}_i = 0$$

```
for(int i = 0; i< NUMBEROFRESOURCES; ++i)
available[i] = atoi(argv[i+1]);

FILE *in;
char *temp;
char customer[MAXLINE];

in = fopen("info.txt","r");

for(int i = 0; i<NUMBEROFCUSTOMERS; ++i){
    if(fgets(customer,MAXLINE,in)==NULL){
        fprintf(stderr,"Read file error!\n");
        return -1;
    }
    temp = strdup(customer);

    for(int j = 0; j<NUMBEROFRESOURCES; ++j){
        int maxres = atoi(strsep(&temp,","));
        need[i][j] = maxres;
        maximum[i][j] = maxres;
        allocation[i][j] = 0;
    }

    free(temp);
}

fclose(in);
```

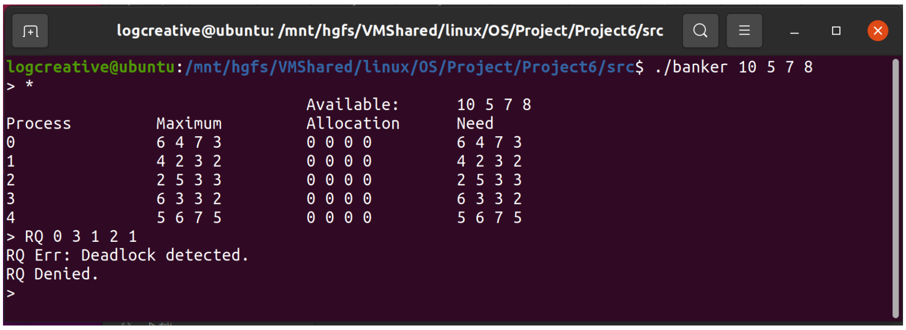
之后，仿照命令行的输入设定输入循环。如果输入 *，则显示系统当前的状态。

```
char command[2];
fprintf(stdout,"> ");
while(fscanf(stdin,"%s",command)!=EOF){
    if (strcmp(command,"RQ")==0){
        int customer_num;
        fscanf(stdin, "%d", &customer_num);
        int request[NUMBEROFRESOURCES];
        for(int j = 0; j < NUMBEROFRESOURCES; ++j)
            fscanf(stdin, "%d", &request[j]);
        if(request_resources(customer_num, request))
            fprintf(stdout, "RQ Denied.\n");
        else fprintf(stdout, "RQ Success.\n");
    } else if (strcmp(command,"RL")==0){
        int customer_num;
        fscanf(stdin, "%d", &customer_num);
        int release[NUMBEROFRESOURCES];
        for(int j = 0; j < NUMBEROFRESOURCES; ++j)
            fscanf(stdin, "%d", &release[j]);
        release_resources(customer_num, release);
        fprintf(stdout, "RL Complete.\n");
    } else if (strcmp(command,"*")==0){
        fprintf(stdout, "\t\t\t\tAvailable: \t");
        for(int j = 0; j<NUMBEROFRESOURCES; ++j)
            fprintf(stdout, "%d ", available[j]);
        fprintf(stdout, "\n");
        fprintf(stdout,"Process\t\tMaximum\t\tAllocation\tNeed\n");
        for(int i = 0; i < NUMBEROFCUSTOMERS; ++i){
            fprintf(stdout, "%d\t\t", i);
            for(int j = 0; j < NUMBEROFRESOURCES; ++j)
                fprintf(stdout, "%d ", maximum[i][j]);
            fprintf(stdout, "\t");
            for(int j = 0; j < NUMBEROFRESOURCES; ++j)
                fprintf(stdout, "%d ", allocation[i][j]);
            fprintf(stdout, "\t");
            for(int j = 0; j < NUMBEROFRESOURCES; ++j)
                fprintf(stdout, "%d ", need[i][j]);
            fprintf(stdout, "\n");
        }
    }
    fprintf(stdout,"> ");
}

return 0;
```

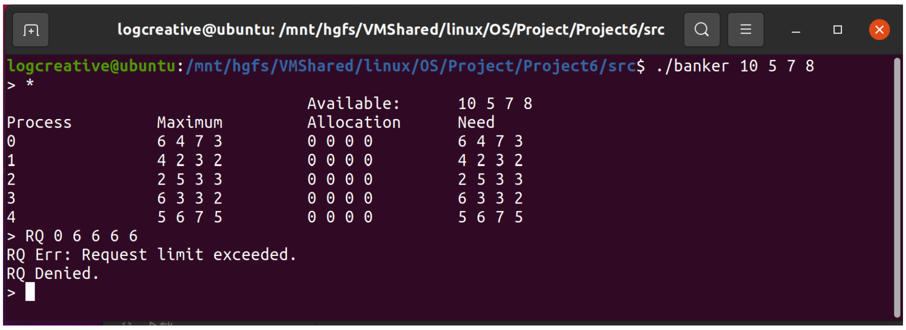
二　请求数据

如果存在死锁，请求将会被拒绝。


logcreative@ubuntu: /mnt/hgfs/VMShared/linux/OS/Project/Project6/src

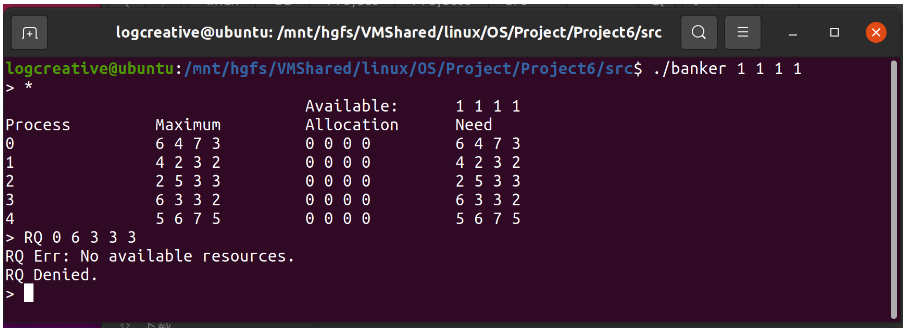
```
logcreative@ubuntu:/mnt/hgfs/VMShared/linux/OS/Project/Project6/src$ ./banker 10 5 7 8
> *
                                Available:      10 5 7 8
Process         Maximum         Allocation      Need
0               6 4 7 3         0 0 0 0         6 4 7 3
1               4 2 3 2         0 0 0 0         4 2 3 2
2               2 5 3 3         0 0 0 0         2 5 3 3
3               6 3 3 2         0 0 0 0         6 3 3 2
4               5 6 7 5         0 0 0 0         5 6 7 5
> RQ 0 3 1 2 1
RQ Err: Deadlock detected.
RQ Denied.
>
```

如果请求了过多的资源，超出了自身的最大值，就会有超限的错误。


logcreative@ubuntu: /mnt/hgfs/VMShared/linux/OS/Project/Project6/src

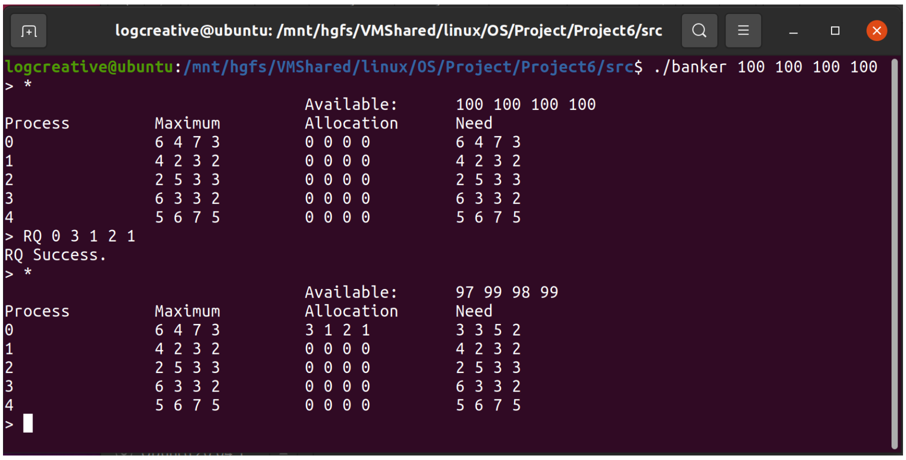
```
logcreative@ubuntu:/mnt/hgfs/VMShared/linux/OS/Project/Project6/src$ ./banker 10 5 7 8
> *
                                Available:      10 5 7 8
Process         Maximum         Allocation      Need
0               6 4 7 3         0 0 0 0         6 4 7 3
1               4 2 3 2         0 0 0 0         4 2 3 2
2               2 5 3 3         0 0 0 0         2 5 3 3
3               6 3 3 2         0 0 0 0         6 3 3 2
4               5 6 7 5         0 0 0 0         5 6 7 5
> RQ 0 6 6 6 6
RQ Err: Request limit exceeded.
RQ Denied.
> 
```

如果请求的资源多于目前可用资源，就会有资源不足的错误。


logcreative@ubuntu: /mnt/hgfs/VMShared/linux/OS/Project/Project6/src

```
logcreative@ubuntu:/mnt/hgfs/VMShared/linux/OS/Project/Project6/src$ ./banker 1 1 1 1
> *
                                Available:      1 1 1 1
Process         Maximum         Allocation      Need
0               6 4 7 3         0 0 0 0         6 4 7 3
1               4 2 3 2         0 0 0 0         4 2 3 2
2               2 5 3 3         0 0 0 0         2 5 3 3
3               6 3 3 2         0 0 0 0         6 3 3 2
4               5 6 7 5         0 0 0 0         5 6 7 5
> RQ 0 6 3 3 3
RQ Err: No available resources.
RQ Denied.
> 
```

如果不存在死锁，请求将会被执行，相关数据将会被刷新。


logcreative@ubuntu: /mnt/hgfs/VMShared/linux/OS/Project/Project6/src

```
logcreative@ubuntu:/mnt/hgfs/VMShared/linux/OS/Project/Project6/src$ ./banker 100 100 100 100
> *
                                Available:      100 100 100 100
Process         Maximum         Allocation      Need
0               6 4 7 3         0 0 0 0         6 4 7 3
1               4 2 3 2         0 0 0 0         4 2 3 2
2               2 5 3 3         0 0 0 0         2 5 3 3
3               6 3 3 2         0 0 0 0         6 3 3 2
4               5 6 7 5         0 0 0 0         5 6 7 5
> RQ 0 3 1 2 1
RQ Success.
> *
                                Available:      97 99 98 99
Process         Maximum         Allocation      Need
0               6 4 7 3         3 1 2 1         3 3 5 2
1               4 2 3 2         0 0 0 0         4 2 3 2
2               2 5 3 3         0 0 0 0         2 5 3 3
3               6 3 3 2         0 0 0 0         6 3 3 2
4               5 6 7 5         0 0 0 0         5 6 7 5
> 
```

请求函数使用了银行家算法。初始时进行平凡情况的判定，以自起始时弹出错误：

```
int request_resources(int customer_num, int request[]){
    for(int j = 0; j < NUMBEROFRESOURCES; ++j){
        if(allocation[customer_num][j] + request[j] > maximum[customer_num][j]){
            fprintf(stderr, "RQ Err: Request limit exceeded.\n");
            return 1;
        }
        if(request[j]>available[j]){
            fprintf(stderr, "RQ Err: No available resources.\n");
            return 1;
        }
    }
```

然后进行假设分配，将可用情况在一个新的数组 allocation_pre 中存储。之后会使用 finish 数组存储任务的完成情况。

```
    // Fake allocation
    int available_pre[NUMBEROFRESOURCES];
    for(int j = 0; j < NUMBEROFRESOURCES; ++j){
        allocation[customer_num][j] += request[j];
        need[customer_num][j] -= request[j];
        available_pre[j] = available[j] - request[j];
    }

    int finish[NUMBEROFCUSTOMERS];
    for(int i = 0; i < NUMBEROFCUSTOMERS; ++i){
        int finish_flag = 1;
        for(int j = 0; j < NUMBEROFRESOURCES; ++j)
            if(need[i][j] > 0)
                finish_flag = 0;
        finish[i] = finish_flag;
    }
```

之后就是标准的银行家算法，如果找到了候选者就会继续循环，否则退出循环。候选者的条件两条需要同时满足：（1）未完成；（2）需要的资源可以被当下的可用资源满足。满足后就会更新可用资源的数据。

```
    int found = 0;
    do {
        found = 0;
        for(int i = 0; i < NUMBEROFCUSTOMERS; ++i){
            if(!finish[i]){
                int next = 1;
                for(int j = 0; j < NUMBEROFRESOURCES; ++j)
                    if(need[i][j] > available_pre[j]){
                        next = 0;
                        break;
                    }
                if(!next) continue;
                finish[i] = 1;
                for(int j = 0; j < NUMBEROFRESOURCES; ++j)
                    available_pre[j] += allocation[i][j];
                found = 1;
                break;
            }
        }
    } while (found);
```

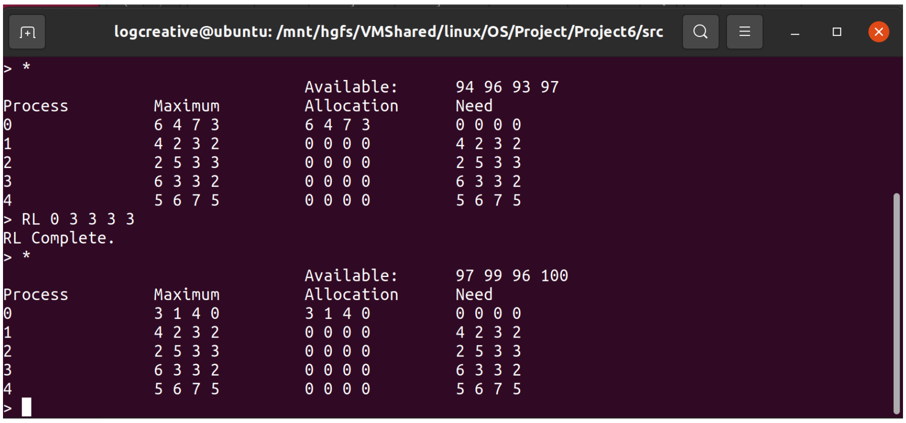
如果退出循环后依然有未完成的任务，则存在死锁，对分配进行回滚操作。如果可行就只需要修改 available 数组的相关值。

```
    int all_finish_flag = 1;
    for(int i = 0; i < NUMBEROFCUSTOMERS; ++i)
        if(!finish[i])
            all_finish_flag = 0;
    if(!all_finish_flag){
        // fallback
        for(int j = 0; j < NUMBEROFRESOURCES; ++j){
            allocation[customer_num][j] -= request[j];
            need[customer_num][j] += request[j];
        }
        fprintf(stderr, "RQ Err: Deadlock detected.\n");
        return 1;
    }

    for(int j = 0; j < NUMBEROFRESOURCES; ++j)
        available[j] -= request[j];
    return 0;
}
```

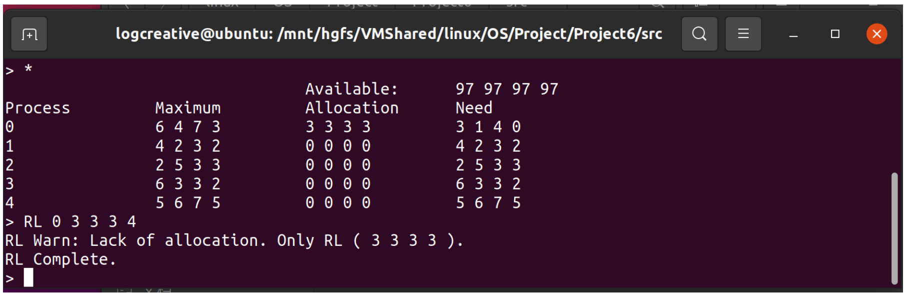
三　资源释放

资源正常释放，Maximum 和 Allocation 的值都会修改。


logcreative@ubuntu: /mnt/hgfs/VMShared/linux/OS/Project/Project6/src

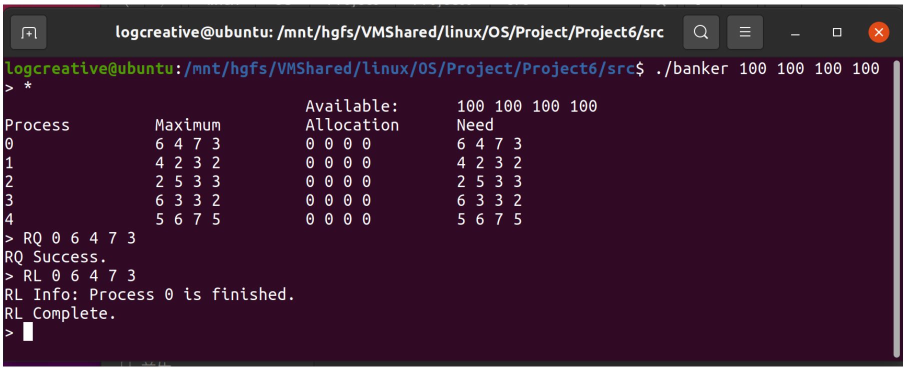
```
> *
                                Available:      94 96 93 97
Process         Maximum         Allocation      Need
0               6 4 7 3         6 4 7 3         0 0 0 0
1               4 2 3 2         0 0 0 0         4 2 3 2
2               2 5 3 3         0 0 0 0         2 5 3 3
3               6 3 3 2         0 0 0 0         6 3 3 2
4               5 6 7 5         0 0 0 0         5 6 7 5
> RL 0 3 3 3 3
RL Complete.
> *
                                Available:      97 99 96 100
Process         Maximum         Allocation      Need
0               3 1 4 0         3 1 4 0         0 0 0 0
1               4 2 3 2         0 0 0 0         4 2 3 2
2               2 5 3 3         0 0 0 0         2 5 3 3
3               6 3 3 2         0 0 0 0         6 3 3 2
4               5 6 7 5         0 0 0 0         5 6 7 5
> 
```

如果释放不了这么多的资源，就会释放该资源能释放的最大值，并弹出警告。


logcreative@ubuntu: /mnt/hgfs/VMShared/linux/OS/Project/Project6/src

```
> *
                                Available:      97 97 97 97
Process         Maximum         Allocation      Need
0               6 4 7 3         3 3 3 3         3 1 4 0
1               4 2 3 2         0 0 0 0         4 2 3 2
2               2 5 3 3         0 0 0 0         2 5 3 3
3               6 3 3 2         0 0 0 0         6 3 3 2
4               5 6 7 5         0 0 0 0         5 6 7 5
> RL 0 3 3 3 4
RL Warn: Lack of allocation. Only RL ( 3 3 3 3 ).
RL Complete.
> 
```

如果资源全部释放完毕，会提示该进程已经结束。


logcreative@ubuntu: /mnt/hgfs/VMShared/linux/OS/Project/Project6/src

```
logcreative@ubuntu:/mnt/hgfs/VMShared/linux/OS/Project/Project6/src$ ./banker 100 100 100 100
> *
                                Available:      100 100 100 100
Process         Maximum         Allocation      Need
0               6 4 7 3         0 0 0 0         6 4 7 3
1               4 2 3 2         0 0 0 0         4 2 3 2
2               2 5 3 3         0 0 0 0         2 5 3 3
3               6 3 3 2         0 0 0 0         6 3 3 2
4               5 6 7 5         0 0 0 0         5 6 7 5
> RQ 0 6 4 7 3
RQ Success.
> RL 0 6 4 7 3
RL Info: Process 0 is finished.
RL Complete.
> 
```

释放函数的定义如下：只要释放就认为这些资源已经不需要了，将会从最大值中减去，该任务不能再请求更多的资源。

```
void release_resources(int customer_num, int release[]){
    int real_release[NUMBEROFRESOURCES];
    int warn_flag = 0;
    for(int j = 0; j < NUMBEROFRESOURCES; ++j){
        if(release[j] > allocation[customer_num][j]){
            real_release[j] = allocation[customer_num][j];
            warn_flag = 1;
        }
        else real_release[j] = release[j];
        allocation[customer_num][j] -= real_release[j];
        maximum[customer_num][j] -= real_release[j];
        available[j] += real_release[j];
    }
    if(warn_flag){
        fprintf(stderr, "RL Warn: Lack of allocation. Only RL ( ");
        for(int j = 0; j < NUMBEROFRESOURCES; ++j)
            fprintf(stdout, "%d ", real_release[j]);
        fprintf(stdout, ").\n");
    }
    int fin_flag = 1;
    for(int j = 0; j < NUMBEROFRESOURCES; ++j)
        if(maximum[customer_num][j]>0)
            fin_flag = 0;
    if(fin_flag)
        fprintf(stdout, "RL Info: Process %d is finished.\n", customer_num);
    }
}
```

A 全部代码

Listing 1: src/Makefile

```
all:
    gcc -g -Wall banker.c -o banker
clean:
    rm *.o banker
```

Listing 2: src/banker.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define NUMBEROFCUSTOMERS 5
#define NUMBEROFRESOURCES 4
#define MAXLINE 1000

/* the available amount of each resource */
int available[NUMBEROFRESOURCES];

/*the maximum demand of each customer */
int maximum[NUMBEROFCUSTOMERS][NUMBEROFRESOURCES];

/* the amount currently allocated to each customer */
int allocation[NUMBEROFCUSTOMERS][NUMBEROFRESOURCES];

/* the remaining need of each customer */
int need[NUMBEROFCUSTOMERS][NUMBEROFRESOURCES];

int request_resources(int customer_num, int request[]){
    for(int j = 0; j < NUMBEROFRESOURCES; ++j){
        if(allocation[customer_num][j] + request[j] > maximum[customer_num][j]){
            fprintf(stderr, "RQ Err: Request limit exceeded.\n");
            return 1;
        }
        if(request[j]>available[j]){
            fprintf(stderr, "RQ Err: No available resources.\n");
            return 1;
        }
    }

    // Fake allocation
    int available_pre[NUMBEROFRESOURCES];
    for(int j = 0; j < NUMBEROFRESOURCES; ++j){
        allocation[customer_num][j] += request[j];
        need[customer_num][j] -= request[j];
        available_pre[j] = available[j] - request[j];
    }

    int finish[NUMBEROFCUSTOMERS];
    for(int i = 0; i < NUMBEROFCUSTOMERS; ++i){
        int finish_flag = 1;
        for(int j = 0; j < NUMBEROFRESOURCES; ++j)
            if(need[i][j] > 0)
                finish_flag = 0;
        finish[i] = finish_flag;
    }

    int found = 0;
    do {
        found = 0;
        for(int i = 0; i < NUMBEROFCUSTOMERS; ++i){
            if(!finish[i]){
                int next = 1;
                for(int j = 0; j < NUMBEROFRESOURCES; ++j)
```

```
                    if(need[i][j] > available_pre[j]){
                        next = 0;
                        break;
                    }
                if(!next) continue;
                finish[i] = 1;
                for(int j = 0; j < NUMBEROFRESOURCES; ++j)
                    available_pre[j] += allocation[i][j];
                found = 1;
                break;
            }
        }
    } while (found);

    int all_finish_flag = 1;
    for(int i = 0; i < NUMBEROFCUSTOMERS; ++i)
        if(!finish[i])
            all_finish_flag = 0;
    if(!all_finish_flag){
        // fallback
        for(int j = 0; j < NUMBEROFRESOURCES; ++j){
            allocation[customer_num][j] -= request[j];
            need[customer_num][j] += request[j];
        }
        fprintf(stderr, "RQ Err: Deadlock detected.\n");
        return 1;
    }

    for(int j = 0; j < NUMBEROFRESOURCES; ++j)
        available[j] -= request[j];
    return 0;
}

void release_resources(int customer_num, int release[]){
    int real_release[NUMBEROFRESOURCES];
    int warn_flag = 0;
    for(int j = 0; j < NUMBEROFRESOURCES; ++j){
        if(release[j] > allocation[customer_num][j]){
            real_release[j] = allocation[customer_num][j];
            warn_flag = 1;
        }
        else real_release[j] = release[j];
        allocation[customer_num][j] -= real_release[j];
        maximum[customer_num][j] -= real_release[j];
        available[j] += real_release[j];
    }
    if(warn_flag){
        fprintf(stderr, "RL Warn: Lack of allocation. Only RL ( ");
        for(int j = 0; j < NUMBEROFRESOURCES; ++j)
            fprintf(stdout, "%d ", real_release[j]);
        fprintf(stdout, ").\n");
    }
    int fin_flag = 1;
    for(int j = 0; j < NUMBEROFRESOURCES; ++j)
        if(maximum[customer_num][j]>0)
            fin_flag = 0;
    if(fin_flag)
        fprintf(stdout, "RL Info: Process %d is finished.\n", customer_num);
}

int main(int argc, char* argv[]){
    for(int i = 0; i< NUMBEROFRESOURCES; ++i)
        available[i] = atoi(argv[i+1]);

    FILE *in;
    char *temp;
    char customer[MAXLINE];

    in = fopen("info.txt","r");

    for(int i = 0; i<NUMBEROFCUSTOMERS; ++i){
        if(fgets(customer,MAXLINE,in)==NULL){
            fprintf(stderr,"Read file error!\n");
            return -1;
        }
        temp = strdup(customer);

        for(int j = 0; j<NUMBEROFRESOURCES; ++j){
            int maxres = atoi(strsep(&temp,","));
            need[i][j] = maxres;
            maximum[i][j] = maxres;
            allocation[i][j] = 0;
        }

        free(temp);
    }

    fclose(in);

    char command[2];
    fprintf(stdout,"> ");
    while(fscanf(stdin,"%s",command)!=EOF){
        if (strcmp(command,"RQ")==0){
            int customer_num;
            fscanf(stdin, "%d", &customer_num);
            int request[NUMBEROFRESOURCES];
            for(int j = 0; j < NUMBEROFRESOURCES; ++j)
                fscanf(stdin, "%d", &request[j]);
            if(request_resources(customer_num, request))
                fprintf(stdout, "RQ Denied.\n");
            else fprintf(stdout, "RQ Success.\n");
        } else if (strcmp(command,"RL")==0){
            int customer_num;
            fscanf(stdin, "%d", &customer_num);
            int release[NUMBEROFRESOURCES];
            for(int j = 0; j < NUMBEROFRESOURCES; ++j)
                fscanf(stdin, "%d", &release[j]);
            release_resources(customer_num, release);
            fprintf(stdout, "RL Complete.\n");
        } else if (strcmp(command,"*")==0){
            fprintf(stdout, "\t\t\t\tAvailable: \t");
            for(int j = 0; j<NUMBEROFRESOURCES; ++j)
```

```
                fprintf(stdout, "%d ", available[j]);
            fprintf(stdout, "\n");
            fprintf(stdout,"Process\t\tMaximum\t\tAllocation\tNeed\n");
            for(int i = 0; i < NUMBEROFCUSTOMERS; ++i){
                fprintf(stdout, "%d\t\t", i);
                for(int j = 0; j < NUMBEROFRESOURCES; ++j)
                    fprintf(stdout, "%d ", maximum[i][j]);
                fprintf(stdout, "\t");
                for(int j = 0; j < NUMBEROFRESOURCES; ++j)
                    fprintf(stdout, "%d ", allocation[i][j]);
                fprintf(stdout, "\t");
                for(int j = 0; j < NUMBEROFRESOURCES; ++j)
                    fprintf(stdout, "%d ", need[i][j]);
                fprintf(stdout, "\n");
            }
        }
        fprintf(stdout,"> ");
    }

    return 0;
}
```

Listing 3: src/info.txt

```
6,4,7,3
4,2,3,2
2,5,3,3
6,3,3,2
5,6,7,5
```