Project 6

秦啸涵 **521021910604**

初始化全局变量

需要声明的全局变量为`available[],need[][],maximum[][],allocation[][]`数组

```
int available[4];
int maximum[5][4];
int need[5][4];
int allocation[5][4];
```

其中`avaliable[]`从命令行作为`main()`函数参数输入，`maximum[][]`数组从文件`input.txt`中读取，`need[][]`数组与`maximum[][]`初始相同，`allocation[][]`数组初始化为0

```
if (argc != 5) {
    printf("invalid available resource.\n");
    return -1;
}
int shouldrun = 1;
char* input;
input = (char*) malloc(MAX_LINE * sizeof(char));
available[0] = atoi(argv[1]);
available[1] = atoi(argv[2]);
available[2] = atoi(argv[3]);
available[3] = atoi(argv[4]);
FILE *fp;
fp = fopen("input.txt", "r");
for (int i=0; i<5; i++)
{
    for (int j=0; j<4; j++)
    {
```

```
            fscanf(fp, "%d", &maximum[i][j]);
            need[i][j] = maximum[i][j];
        }
    }
    fclose(fp);
```

支持指令*打印当前状态

程序循环的主体如下:

```
while(shouldrun)
{
    printf("\033[1;35mbanker>\033[0m");
    fflush(stdout);
    fgets(input, 100, stdin);
    input[strlen(input) - 1] = '\0';
    if (strcmp(input, "exit") == 0) {
        shouldrun = 0;
        continue;
    }
    char* args[10];
    for (int i=0; i<10; i++)
    {
        args[i] = (char*) malloc(10 * sizeof(char));
    }
    int arg_num = parse(input, args);
    if (arg_num == 1 && strcmp(args[0], "*") == 0)
    {
        printstatics();
    }
    if (arg_num == 6 && strcmp(args[0], "RQ") == 0)
    {
        RQ(args);
    }
    if (arg_num == 6 && strcmp(args[0], "RL") == 0)
    {
        RL(args);
    }
}
```

同project 3类似，实现了一个shell的结构，当输入指令为`*`时，调用`printstatics()`函数打印当前状态。

`printstatic()`函数实现如下:

```
void printstatics()
{
    printf("avaliable array is:\n");
    printf("%d %d %d %d\n", available[0], available[1], available[2], available[3]);
    printf("maximum matrix is:\n");
    for (int i = 0; i < 5; i++)
    {
        printf("%d %d %d %d\n", maximum[i][0], maximum[i][1], maximum[i][2], maximum[i][3]);
    }
    printf("allocation matrix is:\n");
    for (int i = 0; i < 5; i++)
    {
        printf("%d %d %d %d\n", allocation[i][0], allocation[i][1], allocation[i][2], allocation[i][3]);
    }
    printf("need matrix is:\n");
    for (int i = 0; i < 5; i++)
    {
        printf("%d %d %d %d\n", need[i][0], need[i][1], need[i][2], need[i][3]);
    }
}
```

运行结果:

```
jianke@ubuntu:~/Desktop/final-src-osc10e/ch8/banker$ ./banker 6 6 7 5
banker>*
avaliable array is:
6 6 7 5
maximum matrix is:
6 4 7 3
4 2 3 2
2 5 3 3
6 3 3 2
5 6 7 5
allocation matrix is:
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0
need matrix is:
6 4 7 3
4 2 3 2
2 5 3 3
6 3 3 2
5 6 7 5
```

支持RQ请求资源

当命令行给出 `RQ` 命令时，会调用 `RQ()` 函数

```
1  void RQ(char **args)
2  {
3      int pid = atoi(args[1]);
4      int request[4];
5      request[0] = atoi(args[2]);
6      request[1] = atoi(args[3]);
7      request[2] = atoi(args[4]);
8      request[3] = atoi(args[5]);
9      for (int i = 0; i < 4; i++)
10     {
11         if (request[i] > need[pid][i])
12         {
13             printf("request is larger than need.\n");
14             return;
15         }
16         if (request[i] > available[i])
17         {
18             printf("request is larger than available.\n");
```

```
            return;
        }
    }
    for (int i = 0; i < 4; i++)
    {
        available[i] -= request[i];
        allocation[pid][i] += request[i];
        need[pid][i] -= request[i];
    }
    if (check_safe())
    {
        printf("\033[32mSuccessfully allocate the resources!\033[0m\n");
    }
    else
    {
        printf("\033[31mThe state is not safe!\033[0m\n");
        for (int i = 0; i < 4; i++)
        {
            available[i] += request[i];
            allocation[pid][i] -= request[i];
            need[pid][i] += request[i];
        }
    }
}
```

`RQ()` 函数首先判断请求的资源是否合法(是否超过了最大可利用资源，是否超过了当前进程需求的资源)，如果请求合法，则会先将资源分配给进程，之后调用 `check_safe()` 函数判断系统是否处于安全态。若处于安全态，则资源分配成功，打印成功信息；否则给出警告，并将资源返回回溯到之前的安全态

```
bool check_safe()
{
    int work[4];
    bool finish[5];
    for (int i = 0; i < 4; i++)
    {
        work[i] = available[i];
    }
```

```
    for (int i = 0; i < 5; i++)
    {
        finish[i] = false;
    }
    int count = 0;
    while (count < 5)
    {
        bool flag = false;
        for (int i = 0; i < 5; i++)
        {
            if (finish[i] == false)
            {
                bool flag2 = true;
                for (int j = 0; j < 4; j++)
                {
                    if (need[i][j] > work[j])
                    {
                        flag2 = false;
                        break;
                    }
                }
                if (flag2)
                {
                    for (int j = 0; j < 4; j++)
                    {
                        work[j] += allocation[i][j];
                    }
                    finish[i] = true;
                    flag = true;
                    count++;
                }
            }
        }
        if (flag == false)
        {
            return false;
        }
    }
    return true;
}
```

运行结果如下(`available[4]={6,6,7,5}`):

```
banker>RQ 0 4 3 3 3
Successfully allocate the resources!
banker>RQ 1 2 2 2 2
The state is not safe!
banker>*
avaliable array is:
2 3 4 2
maximum matrix is:
6 4 7 3
4 2 3 2
2 5 3 3
6 3 3 2
5 6 7 5
allocation matrix is:
4 3 3 3
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0
need matrix is:
2 1 4 0
4 2 3 2
2 5 3 3
6 3 3 2
5 6 7 5
```

支持RL释放资源

当命令行给出 `RL` 命令时，会调用 `RL()` 函数

```
1  void RL(char **args)
2  {
3      int pid = atoi(args[1]);
4      int request[4];
5      request[0] = atoi(args[2]);
6      request[1] = atoi(args[3]);
7      request[2] = atoi(args[4]);
8      request[3] = atoi(args[5]);
9      for (int i = 0; i < 4; i++)
10     {
11         if (request[i] > allocation[pid][i])
12         {
```

```
            printf("\033[31m%d customer doesn't have so many resources!\033[0m\n", pid);
            return;
        }
    }
    for (int i = 0; i < 4; i++)
    {
        available[i] += request[i];
        allocation[pid][i] -= request[i];
        need[pid][i] += request[i];
    }
    printf("\033[32mSuccessfully release the resources!\033[0m\n");
}
```

`RL()` 依然判断当前要释放的资源是否合法，如果要释放的资源数量小于该进程当前已分配的资源数量，则将资源释放并修改 `available[]` 等的值，否则输出错误信息

```
banker>RL 0 4 4 4 4
0 customer doesn't have so many resources!
banker>RL 0 3 0 0 0
Successfully release the resources!
banker>RL 0 1 1 1 1
Successfully release the resources!
banker>*
avaliable array is:
6 4 5 3
maximum matrix is:
6 4 7 3
4 2 3 2
2 5 3 3
6 3 3 2
5 6 7 5
allocation matrix is:
0 2 2 2
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0
need matrix is:
6 2 5 1
4 2 3 2
2 5 3 3
6 3 3 2
5 6 7 5
banker>exit
jianke@ubuntu:~/Desktop/final-src-osc10e/ch8/banker$
```