آزمایشگاه سیستم عامل

گزارش آزمایش شماره 7

نام استاد: مهندس حمیدرضا کیخا

تاریخ: 1400 / 9 / 27

نام و نام خانوادگی: نگار کرمی

شماره دانشجویی: 9722039

نام و نام خانوادگی: مجید نامی

شماره دانشجویی: 9728095

بانکدار درخواست های n مشتری را برای m نوع منبع بررسی خواهد کرد. اما همان طور که در دستور کار وجود دارد برای سادگی ساختمان داده ی زیر را در نظر میگیریم:

```
 8   #define NUMBER_OF_CUSTOMERS 5
 9   #define NUMBER_OF_RESOURCES 5
10
```

```
int finish[NUMBER_OF_CUSTOMERS];

int work[NUMBER_OF_RESOURCES];

int available[NUMBER_OF_RESOURCES];

int maximum[NUMBER_OF_RESOURCES][NUMBER_OF_CUSTOMERS];

int allocation[NUMBER_OF_RESOURCES][NUMBER_OF_CUSTOMERS];

int need[NUMBER_OF_RESOURCES][NUMBER_OF_CUSTOMERS];

pthread_t threads[NUMBER_OF_CUSTOMERS];
```

آرایه available **که مقدار موجود هر منبع است (که همان طور که در دستور کار گفته شده بود برای سادگی ما تعداد منابع را** 5 **در نظر گرفتیم)**

آرایه available **با مقادیر ورودی پر میشود :**

```
    if (argc < NUMBER_OF_RESOURCES + 1)
    {
        printf("not enough arguments\n");
        exit(EXIT_FAILURE);
    }
    for (int i = 0; i < NUMBER_OF_RESOURCES; i++)
    {
        available[i] = strtol(argv[i + 1], NULL, 10);
        work[i] = available[i];
    }
    for (int i = 0; i < NUMBER_OF_RESOURCES; i++)
        printf("av[%d]: %d\n", i, available[i]);

    for (int i = 0; i < NUMBER_OF_CUSTOMERS; i++)
        finish[i] = 0;
```

برای مثال با دستور زیر برنامه خود را با ارسال تعداد هر یک از انواع منابع بر روی خط فرمان احضار می کنیم برای مثال اگر 5 **نوع منبع با** 8 **نمونه از نوع اول،** 3 **نمونه از نوع دوم و** 7 **نمونه از نوع سوم و** 5 **نمونه از نوع چهارم و** 2 **نمونه از نوع پنجم وجود داشته باشد، برنامه خود را یه صورت زیر احضار خواهیم کرد : که آرایه** availabl **به صورت زیر مقدار دهی میشود :**

```
majid@majid-virtual-machine:~/Desktop/L7$ gcc main.c -lpthread -o main; ./main 8 3 7 5 2
av[0]: 8
av[1]: 3
av[2]: 7
av[3]: 5
av[4]: 2
```

روش مناسب برای مقداردهی آرایه maximum بیابید :

حداکثر تقاضای مشتری که همان طور که در دستورکار گفته شده است ، تعدادی تصادفی از منابع درخواست میشود

پس برای اینکه آرایه maximum را به صورت خواسته شده پر کنیم از روش زیر استفاده میکنیم :

```
for (int i = 0; i < NUMBER_OF_RESOURCES; i++)
{
    for (int j = 0; j < NUMBER_OF_CUSTOMERS; j++)
    {
        allocation[i][j] = 0;
        maximum[i][j] = (rand() % available[i]) + 1;
        need[i][j] = maximum[i][j];
        printf("%d ", need[i][j]);
    }
    printf("\n");
}
```

برای مثال در مثالی که در بالا زده شد آرایه maximum در دو اجرای مختلف به صورت های زیر ایجاد میشود:

```
majid@majid-virtual-machine:~/Desktop/L7$ gcc main.c -lpthread -o main; ./main 8 3 7 5 2
av[0]: 8
av[1]: 3
av[2]: 7
av[3]: 5
av[4]: 2
allocation is zero !!!
maximum = need :
8 4 4 8 8
3 3 3 3 1
4 2 1 1 3
5 3 4 5 1
1 1 1 1 1
```

```
majid@majid-virtual-machine:~/Desktop/L7$ gcc main.c -lpthread -o main; ./main 8 3 7 5 2
av[0]: 8
av[1]: 3
av[2]: 7
av[3]: 5
av[4]: 2
allocation is zero !!!
maximum = need :
8 6 4 1 4
2 1 1 2 3
1 6 5 1 5
3 1 1 5 5
1 2 2 1 2
```

مطابق کد زیر اگر در حالت امن باشیم thread ها ساخته میشوند و اگر در حالت ناامن باشیم خارج میشویم : (ما یک حالت امن داریم که اگر زمانی که فرایند جدید وارد سیستم میشود ، در حالت امن باشیم ، فرایند جدید پذیرفته میشود و اجرای آن شروع میشود)

```c
if (state == SAFE)
{
    printf("we are in the safe state !!!\n");
    printf("going to start working with clients\n");
    int indexes[NUMBER_OF_CUSTOMERS];
    for (size_t i = 0; i < NUMBER_OF_CUSTOMERS; i++)
    {
        indexes[i] = i;
    }

    for (int i = 0; i < NUMBER_OF_CUSTOMERS; i++)
    {
        pthread_create(&threads[i], NULL, customer_handler, (void *)(&indexes[i]));
    }
    for (int i = 0; i < NUMBER_OF_CUSTOMERS; i++)
        pthread_join(threads[i], NULL);
}
else
{
    printf("unsafe state \n");
    exit(EXIT_FAILURE);
}
```

مطابق شبه کدی که در دستور کار داشتیم ، بررسی امن بودن مطابق کد زیر صورت میگیرد :

```
SystemState safe()
{
    for (int i = 0; i < NUMBER_OF_CUSTOMERS; i++)
    {
        if (!finish[i] && ls_eq(need[i], work, NUMBER_OF_RESOURCES))
        {
            for (int j = 0; j < NUMBER_OF_RESOURCES; j++)
                work[j] = work[j] + allocation[i][j];
            finish[i] = 1;
        }
    }
    SystemState is_safe = SAFE;
    for (int i = 0; i < NUMBER_OF_CUSTOMERS; i++)
        if (!finish[i])
            is_safe = UNSAFE;

    return is_safe;
}
```

همچنین در دستور کار دو تابع برای درخواست و پس دادن منابع معرفی شده بودند که به صورت زیر پیاده سازی شدند :

تابع request_resources **:**

```c
int request_resource(int customer_num, int *request)
{
    if (!ls_eq(request, need[customer_num], NUMBER_OF_RESOURCES))
    {
        exit(EXIT_FAILURE);
    }

    pthread_mutex_lock(&lock);

    printf("request:{ ");
    for (size_t i = 0; i < NUMBER_OF_RESOURCES; i++)
        printf("%d ", request[i]);
    printf("} from customer: %d\n", customer_num);

    while (!ls_eq(request, available, NUMBER_OF_RESOURCES))
        pthread_cond_wait(&cond, &lock);

    printf("--------------------\n");
    printf("request from customer %d is ok \n", customer_num);

    for (int i = 0; i < NUMBER_OF_RESOURCES; i++)
        available[i] = available[i] - request[i];

    for (int i = 0; i < NUMBER_OF_RESOURCES; i++)
        allocation[customer_num][i] = allocation[customer_num][i] + request[i];

    for (int i = 0; i < NUMBER_OF_RESOURCES; i++)
        need[customer_num][i] = need[customer_num][i] - request[i];

    printf("available: \n");
    for (size_t i = 0; i < NUMBER_OF_RESOURCES; i++)
    {
        printf("%d ", available[i]);
    }
    printf("\n");
    printf("--------------------\n");
    pthread_mutex_unlock(&lock);
    return 0;
}
```

تابع release_resources :

```c
int release_resource(int customer_num, int *request)
{
    pthread_mutex_lock(&lock);
    printf("request from customer %d released resources.\n", customer_num);

    for (int i = 0; i < NUMBER_OF_RESOURCES; i++)
        available[i] = available[i] + request[i];

    for (int i = 0; i < NUMBER_OF_RESOURCES; i++)
        allocation[customer_num][i] = allocation[customer_num][i] - request[i];


    printf("need:{ ");
    for (size_t i = 0; i < NUMBER_OF_RESOURCES; i++)
        printf("%d ", need[customer_num][i]);
    printf("} from customer: %d\n", customer_num);

    pthread_mutex_unlock(&lock);
    pthread_cond_signal(&cond);

    return 1;
}
```

برای مثال اگر برنامه را با 5 نوع منبع با 100 نمونه از نوع اول، 95 نمونه از نوع دوم و 100 نمونه از نوع سوم و 95 نمونه از نوع چهارم و 99 نمونه از نوع پنجم اجرا کنیم

خروجی برنامه ما و انجام عملیات ها به صورت زیر خواهد بود:

```
majid@majid-virtual-machine:~/Desktop/L7$ gcc main.c -lpthread -o main; ./main 100 95 100 95  99
av[0]: 100
av[1]: 95
av[2]: 100
av[3]: 95
av[4]: 99
allocation is zero !!!
maximum = need :
87 10 40 23 45
6 92 91 68 68
48 50 69 91 58
60 66 29 59 7
56 68 98 92 31
we are in the safe state !!!
going to start working with clients
request:{ 47 40 59 81 30 } from customer: 2
--------------------
request from customer 2 is ok
available:
53 55 41 14 69
--------------------
request:{ 50 9 19 81 13 } from customer: 4
request:{ 2 83 24 39 30 } from customer: 1
request:{ 24 48 26 57 3 } from customer: 3
request:{ 41 4 7 16 1 } from customer: 0
request from customer 2 released resources.
need:{ 1 10 10 10 28 } from customer: 2
request:{ 0 9 8 0 19 } from customer: 2
--------------------
request from customer 2 is ok
available:
100 86 92 95 80
--------------------
--------------------
request from customer 4 is ok
available:
50 77 73 14 67
--------------------
request from customer 2 released resources.
need:{ 1 1 2 10 9 } from customer: 2
request:{ 0 1 0 5 6 } from customer: 2
--------------------
request from customer 2 is ok
available:
```

```
--------------------
request from customer 2 released resources.
need:{ 1 1 2 10 9 } from customer: 2
request:{ 0 1 0 5 6 } from customer: 2
--------------------
request from customer 2 is ok
available:
50 85 81 9 80
--------------------
request from customer 4 released resources.
need:{ 6 59 79 11 18 } from customer: 4
--------------------
request from customer 3 is ok
available:
76 46 74 33 90
--------------------
request:{ 5 30 60 4 17 } from customer: 4
--------------------
request from customer 4 is ok
available:
71 16 14 29 73
--------------------
request from customer 3 released resources.
need:{ 36 18 3 2 4 } from customer: 3
request:{ 25 0 2 0 1 } from customer: 3
--------------------
request from customer 3 is ok
available:
70 64 38 86 75
--------------------
request from customer 2 released resources.
need:{ 1 0 2 5 3 } from customer: 2
request from customer 4 released resources.
need:{ 1 29 19 7 1 } from customer: 4
request:{ 1 0 2 2 0 } from customer: 2
--------------------
request from customer 2 is ok
available:
74 95 96 93 98
--------------------
--------------------
```

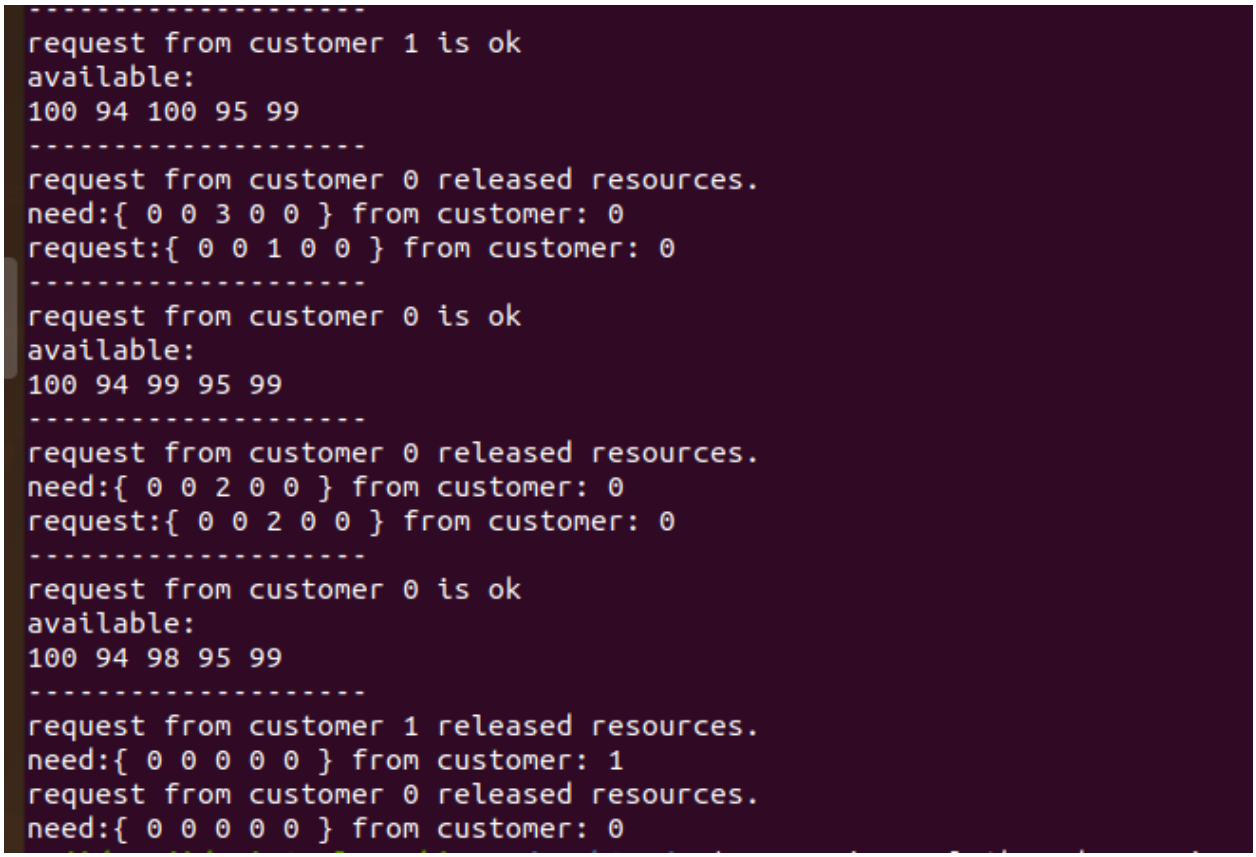
```
--------------------
--------------------
request from customer 0 is ok
available:
33 91 89 77 97
--------------------
--------------------
request from customer 1 is ok
available:
31 8 65 38 67
--------------------
request:{ 1 9 8 7 1 } from customer: 4
request from customer 3 released resources.
need:{ 11 18 1 2 3 } from customer: 3
request:{ 7 0 1 1 1 } from customer: 3
--------------------
request from customer 3 is ok
available:
49 8 66 37 67
--------------------
request from customer 2 released resources.
need:{ 0 0 0 3 3 } from customer: 2
request:{ 0 0 0 2 2 } from customer: 2
--------------------
request from customer 2 is ok
available:
50 8 68 37 65
--------------------
request from customer 1 released resources.
need:{ 4 9 67 29 38 } from customer: 1
request:{ 0 2 48 15 33 } from customer: 1
--------------------
request from customer 1 is ok
available:
52 89 44 61 62
--------------------
request from customer 0 released resources.
need:{ 46 6 33 7 44 } from customer: 0
request:{ 46 0 4 4 40 } from customer: 0
--------------------
request from customer 0 is ok
```

```
--------------------
request from customer 0 is ok
available:
47 93 47 73 23
--------------------
--------------------
request from customer 4 is ok
available:
46 84 39 66 22
--------------------
request from customer 2 released resources.
need:{ 0 0 0 1 1 } from customer: 2
request:{ 0 0 0 1 1 } from customer: 2
--------------------
request from customer 2 is ok
available:
46 84 39 67 23
--------------------
request from customer 1 released resources.
need:{ 4 7 19 14 5 } from customer: 1
request:{ 3 4 15 3 5 } from customer: 1
--------------------
request from customer 1 is ok
available:
43 82 72 79 51
--------------------
request from customer 3 released resources.
need:{ 4 18 0 1 2 } from customer: 3
request:{ 0 14 0 1 1 } from customer: 3
--------------------
request from customer 3 is ok
available:
50 68 73 79 51
--------------------
request from customer 4 released resources.
need:{ 0 20 11 0 0 } from customer: 4
request:{ 0 2 11 0 0 } from customer: 4
--------------------
request from customer 4 is ok
available:
51 75 70 86 52
--------------------
```

```
--------------------
request from customer 0 released resources.
need:{ 0 6 29 3 4 } from customer: 0
request:{ 0 6 16 2 2 } from customer: 0
--------------------
request from customer 0 is ok
available:
97 69 58 88 90
--------------------
request from customer 2 released resources.
need:{ 0 0 0 0 0 } from customer: 2
request from customer 3 released resources.
need:{ 4 4 0 0 1 } from customer: 3
request:{ 3 2 0 0 0 } from customer: 3
--------------------
request from customer 3 is ok
available:
94 81 58 90 92
--------------------
request from customer 1 released resources.
need:{ 1 3 4 11 0 } from customer: 1
request:{ 0 2 4 10 0 } from customer: 1
--------------------
request from customer 1 is ok
available:
97 83 69 83 97
--------------------
request from customer 4 released resources.
need:{ 0 18 0 0 0 } from customer: 4
request:{ 0 16 0 0 0 } from customer: 4
--------------------
request from customer 4 is ok
available:
97 69 80 83 97
--------------------
request from customer 0 released resources.
need:{ 0 0 13 1 2 } from customer: 0
request:{ 0 0 9 0 0 } from customer: 0
--------------------
request from customer 0 is ok
available:
97 75 87 85 99
--------------------
```

```
--------------------
request from customer 3 released resources.
need:{ 1 2 0 0 1 } from customer: 3
request:{ 1 2 0 0 1 } from customer: 3
--------------------
request from customer 3 is ok
available:
99 75 87 85 98
--------------------
request from customer 1 released resources.
need:{ 1 1 0 1 0 } from customer: 1
request:{ 1 0 0 1 0 } from customer: 1
--------------------
request from customer 1 is ok
available:
98 77 91 94 98
--------------------
request from customer 4 released resources.
need:{ 0 2 0 0 0 } from customer: 4
request:{ 0 2 0 0 0 } from customer: 4
--------------------
request from customer 4 is ok
available:
98 91 91 94 98
--------------------
request from customer 0 released resources.
need:{ 0 0 4 1 2 } from customer: 0
request:{ 0 0 0 1 2 } from customer: 0
--------------------
request from customer 0 is ok
available:
98 91 100 93 96
--------------------
request from customer 3 released resources.
need:{ 0 0 0 0 0 } from customer: 3
request from customer 1 released resources.
need:{ 0 1 0 0 0 } from customer: 1
request:{ 0 0 0 0 0 } from customer: 1
--------------------
```

```
--------------------
request from customer 3 released resources.
need:{ 0 0 0 0 0 } from customer: 3
request from customer 1 released resources.
need:{ 0 1 0 0 0 } from customer: 1
request:{ 0 0 0 0 0 } from customer: 1
--------------------
request from customer 1 is ok
available:
100 93 100 94 97
--------------------
request from customer 4 released resources.
need:{ 0 0 0 0 0 } from customer: 4
request from customer 0 released resources.
need:{ 0 0 4 0 0 } from customer: 0
request:{ 0 0 1 0 0 } from customer: 0
--------------------
request from customer 0 is ok
available:
100 95 99 95 99
--------------------
request from customer 1 released resources.
need:{ 0 1 0 0 0 } from customer: 1
request:{ 0 0 0 0 0 } from customer: 1
--------------------
request from customer 1 is ok
available:
100 95 99 95 99
--------------------
request from customer 0 released resources.
need:{ 0 0 3 0 0 } from customer: 0
request:{ 0 0 0 0 0 } from customer: 0
--------------------
request from customer 0 is ok
available:
100 95 100 95 99
--------------------
request from customer 1 released resources.
need:{ 0 1 0 0 0 } from customer: 1
request:{ 0 1 0 0 0 } from customer: 1
--------------------
```


request from customer 1 is ok
available:
100 94 100 95 99
request from customer 0 released resources.
need:{ 0 0 3 0 0 } from customer: 0
request:{ 0 0 1 0 0 } from customer: 0
request from customer 0 is ok
available:
100 94 99 95 99
request from customer 0 released resources.
need:{ 0 0 2 0 0 } from customer: 0
request:{ 0 0 2 0 0 } from customer: 0
request from customer 0 is ok
available:
100 94 98 95 99
request from customer 1 released resources.
need:{ 0 0 0 0 0 } from customer: 1
request from customer 0 released resources.
need:{ 0 0 0 0 0 } from customer: 0

همچنین برای مثال اگر برنامه را با ورودی های زیر اجرا کنیم در ابتدا وارد حالت ناامن خواهیم شد :