**به نام خدا**

# گزارش آزمایش هفتم آزمایشگاه سیستم های عامل

**زهرا رحیمی**

**شماره دانشجوئی: ۹۸۳۱۰۲۶**

**استاد آزمایشگاه: سرکار خانم حسینی**

**پاییز ۱۴۰۰**

در این آزمایش سعی داشتیم برای جلوگیری از بن بست الگوریتم بانک داران را پیاده سازی کنیم. برای این کار ابتدا چند آرایه برای نشان دادن منابع مورد دسترس فرآیند ها (available) و ماکسیمم تعداد نمونه هایی که یک فرآیند می تواند از یک منبع بردارد (max) و تعداد نمونه هایی که فرآیند از یک منبع را در اختیار دارد(allocation) و نمونه های منابع مورد نیاز فرآیند ها(need) را تعریف کرده ایم.

توابع request_resources که در داخل آن هم یک mutex برای اتمیک اجرا شدن آن قرار داده ایم، برای درخواست نمونه ای از منبع و تابع release_request هم برای آزاد سازی نمونه ای از منبع تعریف شده اند.

Customer_thread در واقع یکی از آرگومان های ساخت رشته های کاربران(customer) است و در آن با رندوم کردن درخواست/آزادسازی و همین طور مقدار دهی request ها به طور رندوم از دو تابع request_resources و release_request بهره برده ایم و در هر مرحله نتیجه را به نمایش گذاشته ایم. Safety_algorithm تابعی است که چک میکند آیا با مقادیر حاضر need و available ممکن است بن بست پیش آید یا خیر.

```
bool safety_algorithm();
void * customer_thread(int n);
int request_resources(int customer_num, int request[]);
int release_resources(int customer_num, int request[]);

#define NUMBER_OF_RESOURCES 8
#define NUMBER_OF_CUSTOMERS 10
#define MAX_RESOURCES 20

int available[NUMBER_OF_RESOURCES];
int max[NUMBER_OF_CUSTOMERS][NUMBER_OF_RESOURCES];
int allocation[NUMBER_OF_CUSTOMERS][NUMBER_OF_RESOURCES];
int need[NUMBER_OF_CUSTOMERS][NUMBER_OF_RESOURCES];
```

توابع request_resources و release_resources:

```
int request_resources(int customer_num, int request[]){
    bool flag = true;
    for (int j = 0; j < NUMBER_OF_RESOURCES; ++j) {
        if (request[j] > need[customer_num][j]) {
            perror("Process has exceeded its max");
            return -1;
        }

    }
    sem_wait(&mutex);

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

    if (safety_algorithm()){
        sem_post(&mutex);
        return 0;
    } else {
        for (int j = 0; j < NUMBER_OF_RESOURCES; ++j) {
            available[j] += request[j];
            allocation[customer_num][j] -=request[j];
            need[customer_num][j] += request[j];
        }
    }
    sem_post(&mutex);
    return -1;
}
int release_resources(int customer_num, int request[]){
    sem_wait(&mutex);
    for (int j = 0; j < NUMBER_OF_RESOURCES; ++j) {
        available[j] += request[j];
        allocation[customer_num][j] -=request[j];
        need[customer_num][j] += request[j];
    }
    sem_post(&mutex);
return 1;
}
```

تابع safety_algorithm:

```
bool safety_algorithm(){
    int work[NUMBER_OF_RESOURCES];
    bool finish[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++) {
        finish[i] = false;
    }

    for (int i = 0; i < NUMBER_OF_CUSTOMERS; ++i) {
        if (!finish[i]){
            bool flag = true;
            for (int j = 0; j < NUMBER_OF_RESOURCES; ++j) {
                if (need[i][j] > work[j]){
                    flag = false;
                }
            }
                if(flag) {
                        for (int j = 0; j < NUMBER_OF_RESOURCES; ++j) {

                            work[i] = work[i] + allocation[i][j];

                        }
                        finish[i] = true;
                }

        }
    }
    for (int k = 0; k < NUMBER_OF_CUSTOMERS; ++k) {
        if (!finish[k]) {
            return false;
        }
    }

    return true;

}
```

## چگونگی چند نخی شدن فرآیند مشتریان:

```
pthread_t customer_threads[NUMBER_OF_CUSTOMERS];

    for (int k = 0; k < NUMBER_OF_CUSTOMERS; ++k) {
        pthread_create(&customer_threads[k], NULL, (void *) customer_thread, (void *) k);
    }
    for (int i = 0; i < NUMBER_OF_CUSTOMERS; ++i) {
        pthread_join(customer_threads[i], NULL);
    }


void * customer_thread(int n){
    srand(time(NULL));
    int request[NUMBER_OF_RESOURCES];
for (int i = 0; i < NUMBER_OF_CUSTOMERS; ++i) {
    int type = rand() % 2;
    if (type == 0) {
        for (int j = 0; j < NUMBER_OF_RESOURCES; ++j) {
            request[j] = rand() % (need[n][j]+1);
        }

        printf("Costumer %d has requested %s\b \n\t" , n, request);
        printf("\tResult: %s", request_resources(n, request) ==0 ? "Accepted -> Resource Allocated":
"Not Accepted : Resource Not Allocated\n");

    } else {
        for (int j = 0; j < NUMBER_OF_RESOURCES; ++j) {
            request[j] = rand() % (allocation[n][j]+1);
        }
        release_resources(n, request);
printf("Costumer %d has released %s\n" , n, request);
    }
}

}
```

خروجی آزمایش به صورت زیر است:

```
zahra@zahra-virtual-machine:~/Desktop/OS_Lab/az 7$ gcc bankers.c -pthread -o bankers
bankers.c: In function ‘main’:
bankers.c:95:78: warning: cast to pointer from integer of different size [-Wint-to-pointer-cast]
        pthread_create(&customer_threads[k], NULL, (void *) customer_thread, (void *) k);
                                                                             ^
bankers.c: In function ‘customer_thread’:
bankers.c:196:37: warning: format ‘%s’ expects argument of type ‘char *’, but argument 3 has type ‘int *’ [-Wformat=]
  printf("Costumer %d has requested %s\b \n\t" , n, request);
                                    ~^            ~~~~~~~
                                     %ls
bankers.c:204:35: warning: format ‘%s’ expects argument of type ‘char *’, but argument 3 has type ‘int *’ [-Wformat=]
 printf("Costumer %d has released %s\n" , n, request);
                                  ~^       ~~~~~~~
                                   %ls
zahra@zahra-virtual-machine:~/Desktop/OS_Lab/az 7$ ./bankers 2 3 4 5 6 4 1
Available Array:
av[0]: 2
av[1]: 3
av[2]: 4
av[3]: 5
av[4]: 6

Maximum:
max[0]: 3       max[0]: 6       max[0]: 17      max[0]: 15      max[0]: 13
max[1]: 15      max[1]: 6       max[1]: 12      max[1]: 9       max[1]: 1
max[2]: 2       max[2]: 7       max[2]: 10      max[2]: 19      max[2]: 3
max[3]: 6       max[3]: 0       max[3]: 6       max[3]: 12      max[3]: 16
max[4]: 11      max[4]: 8       max[4]: 7       max[4]: 9       max[4]: 2
max[5]: 10      max[5]: 2       max[5]: 3       max[5]: 7       max[5]: 15
Allocation:
alloc[0]: 0     alloc[0]: 0     alloc[0]: 0     alloc[0]: 0     alloc[0]: 0
alloc[1]: 0     alloc[1]: 0     alloc[1]: 0     alloc[1]: 0     alloc[1]: 0
alloc[2]: 0     alloc[2]: 0     alloc[2]: 0     alloc[2]: 0     alloc[2]: 0
alloc[3]: 0     alloc[3]: 0     alloc[3]: 0     alloc[3]: 0     alloc[3]: 0
alloc[4]: 0     alloc[4]: 0     alloc[4]: 0     alloc[4]: 0     alloc[4]: 0
alloc[5]: 0     alloc[5]: 0     alloc[5]: 0     alloc[5]: 0     alloc[5]: 0
Need:
need[0]: 3      need[0]: 6      need[0]: 17     need[0]: 15     need[0]: 13
need[1]: 15     need[1]: 6      need[1]: 12     need[1]: 9      need[1]: 1
need[2]: 2      need[2]: 7      need[2]: 10     need[2]: 19     need[2]: 3
need[3]: 6      need[3]: 0      need[3]: 6      need[3]: 12     need[3]: 16
need[4]: 11     need[4]: 8      need[4]: 7      need[4]: 9      need[4]: 2
need[5]: 10     need[5]: 2      need[5]: 3      need[5]: 7      need[5]: 15
```

```
Costumer 0 has requested
                Result: Not Accepted : Resource Not Allocated
Costumer 0 has requested
                Result: Not Accepted : Resource Not Allocated
Costumer 0 has requested
                Result: Not Accepted : Resource Not Allocated
Costumer 0 has released
Costumer 0 has requested
                Result: Not Accepted : Resource Not Allocated
Costumer 0 has released
Costumer 1 has requested
                Result: Not Accepted : Resource Not Allocated
Costumer 5 has requested
        Costumer 2 has requested
                Result: Not Accepted : Resource Not Allocated
Costumer 2 has requested
                Result: Not Accepted : Resource Not Allocated
Costumer 2 has requested
        Costumer 4 has requested
                Result: Not Accepted : Resource Not Allocated
Costumer 4 has requested
                Result: Not Accepted : Resource Not Allocated
Costumer 4 has requested
                Result: Not Accepted : Resource Not Allocated
Costumer 4 has released
Costumer 4 has requested
                Result: Not Accepted : Resource Not Allocated
Costumer 4 has released
        Result: Not Accepted : Resource Not Allocated
Costumer 5 has released
Costumer 5 has released
Costumer 5 has released
Costumer 5 has requested
                Result: Not Accepted : Resource Not Allocated
Costumer 5 has released
Costumer 3 has requested
                Result: Not Accepted : Resource Not Allocated
Costumer 3 has requested
                Result: Not Accepted : Resource Not Allocated
Costumer 3 has requested
                Result: Not Accepted : Resource Not Allocated
Costumer 3 has released
```

```
Costumer 2 has released
Costumer 2 has released
Costumer 1 has requeste
                Result: Not Accepted : Resource Not Allocated
Costumer 1 has requested
                Result: Not Accepted : Resource Not Allocated
Costumer 1 has released
Costumer 1 has requested
                Result: Not Accepted : Resource Not Allocated
Costumer 1 has released
```