دستور کار آزمایشگاه سیستم های عامل

آزمایشگاه سیستم های عامل

آرین محسنی

کاوه احمدی

آزمایش ۱۰

دی 1403

دستور کار آزمایشگاه سیستم های عامل

در این بخش الگوریتم بانکدار را با 5 منبع و 5 مشتری اجرا میکنیم.

```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <stdbool.h>

#define NUMBER_OF_RESOURCES 5
#define NUMBER_OF_CUSTOMERS 5

// Shared data structures
int available[NUMBER_OF_RESOURCES];                    // Available resources
int maximum[NUMBER_OF_CUSTOMERS][NUMBER_OF_RESOURCES];   // Maximum demand of
each customer
int allocation[NUMBER_OF_CUSTOMERS][NUMBER_OF_RESOURCES]; // Allocated resources
int need[NUMBER_OF_CUSTOMERS][NUMBER_OF_RESOURCES];       // Remaining needs

pthread_mutex_t lock; // Mutex for synchronization
// Function prototypes
int request_resources(int customer_num, int request[]);
int release_resources(int customer_num, int request[]);
bool is_safe();

void *customer_thread(void *arg) {
    int customer_num = *((int *)arg);
    int request[NUMBER_OF_RESOURCES];
    while (1) {
        // Generate a random resource request
        pthread_mutex_lock(&lock);
        for (int i = 0; i < NUMBER_OF_RESOURCES; i++) {
            request[i] = rand() % (need[customer_num][i] + 1); // Request within
need
        }
        pthread_mutex_unlock(&lock);
        // Attempt to request resources
        if (request_resources(customer_num, request) == 0) {
            printf("Customer %d's request granted\n", customer_num);

            // Simulate resource usage
            sleep(1);

            // Release resources
```

دستور کار آزمایشگاه سیستم های عامل

```
            release_resources(customer_num, request);
            printf("Customer %d released resources\n", customer_num);
        } else {
            printf("Customer %d's request denied\n", customer_num);
        }
        sleep(1);
    }
    return NULL;
}

int request_resources(int customer_num, int request[]) {
    pthread_mutex_lock(&lock);

    // Check if request exceeds need or available resources
    for (int i = 0; i < NUMBER_OF_RESOURCES; i++) {
        if (request[i] > need[customer_num][i] || request[i] > available[i]) {
            pthread_mutex_unlock(&lock);
            return -1; // Request denied
        }
    }
    // Tentatively allocate resources
    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];
    }
    // Check system safety
    if (!is_safe()) {
        // Rollback allocation
        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];
        }
        pthread_mutex_unlock(&lock);
        return -1; // Request denied
    }

    pthread_mutex_unlock(&lock);
    return 0; // Request granted
}

int release_resources(int customer_num, int request[]) {
    pthread_mutex_lock(&lock);
```

دستور کار آزمایشگاه سیستم های عامل

```
    // Release resources
    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];
    }
    pthread_mutex_unlock(&lock);
    return 0;
}
bool is_safe() {
    int work[NUMBER_OF_RESOURCES];
    bool finish[NUMBER_OF_CUSTOMERS] = {false};

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

    // Check safety
    while (true) {
        bool progress = false;
        for (int i = 0; i < NUMBER_OF_CUSTOMERS; i++) {
            if (!finish[i]) {
                bool can_allocate = true;
                for (int j = 0; j < NUMBER_OF_RESOURCES; j++) {
                    if (need[i][j] > work[j]) {
                        can_allocate = false;
                        break;
                    }
                }
                if (can_allocate) {
                    for (int j = 0; j < NUMBER_OF_RESOURCES; j++) {
                        work[j] += allocation[i][j];
                    }
                    finish[i] = true;
                    progress = true;
                }
            }
        }
        if (!progress) break;
    }

    // If all customers are finished, the system is safe
```

دستور کار آزمایشگاه سیستم های عامل

```
    for (int i = 0; i < NUMBER_OF_CUSTOMERS; i++) {
        if (!finish[i]) return false;
    }
    return true;
}
int main(int argc, char *argv[]) {
    if (argc != NUMBER_OF_RESOURCES + 1) {
        printf("Usage: %s <resource1> <resource2> ... <resource%d>\n", argv[0],
NUMBER_OF_RESOURCES);
        return EXIT_FAILURE;
    }
    // Initialize available resources
    for (int i = 0; i < NUMBER_OF_RESOURCES; i++) {
        available[i] = strtol(argv[i + 1], NULL, 10);
    }
    // Randomly initialize maximum and need arrays
    srand(time(NULL));
    for (int i = 0; i < NUMBER_OF_CUSTOMERS; i++) {
        for (int j = 0; j < NUMBER_OF_RESOURCES; j++) {
            maximum[i][j] = rand() % (available[j] + 1);
            allocation[i][j] = 0;
            need[i][j] = maximum[i][j];
        }
    }
    pthread_mutex_init(&lock, NULL);
    // Create customer threads
    pthread_t threads[NUMBER_OF_CUSTOMERS];
    int customer_ids[NUMBER_OF_CUSTOMERS];
    for (int i = 0; i < NUMBER_OF_CUSTOMERS; i++) {
        customer_ids[i] = i;
        pthread_create(&threads[i], NULL, customer_thread, &customer_ids[i]);
    }

    // Join threads
    for (int i = 0; i < NUMBER_OF_CUSTOMERS; i++) {
        pthread_join(threads[i], NULL);
    }

    pthread_mutex_destroy(&lock);
    return 0;
}
```

دستور کار آزمایشگاه سیستم های عامل

این برنامه، الگوریتم بانکداران را برای مدیریت تخصیص منابع در یک سیستم چندنخی پیاده‌سازی می‌کند. مشتریان به صورت نخی منابع را درخواست و آزاد می‌کنند، و قفل‌های انحصار متقابل از شرایط مسابقه جلوگیری می‌کنند. درخواست منابع فقط در صورت باقی‌ماندن سیستم در حالت ایمن پذیرفته می‌شود، و در غیر این صورت رد می‌گردد. الگوریتم ایمنی وضعیت سیستم را بررسی کرده و از وقوع بن‌بست جلوگیری می‌کند. مقادیر اولیه منابع از خط فرمان دریافت شده و تخصیص به صورت پویا شبیه‌سازی می‌شود.

خروجی: با مقادیر 6 8 7 5 10

```
Customer 3's request granted
Customer 2's request denied
Customer 4's request denied
Customer 1's request granted
Customer 0's request granted
Customer 3 released resources
Customer 2's request denied
Customer 4's request denied
Customer 1 released resources
Customer 0 released resources
Customer 3's request granted
Customer 2's request denied
Customer 4's request denied
Customer 1's request denied
Customer 0's request denied
Customer 0's request denied
Customer 3 released resources
Customer 2's request granted
Customer 4's request denied
Customer 1's request granted
Customer 3's request denied
Customer 2 released resources
Customer 4's request denied
Customer 1 released resources
```