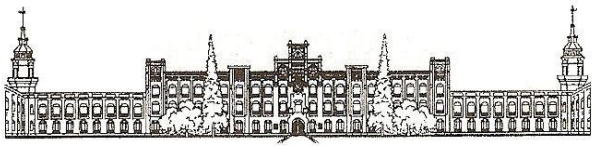


Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут»

Лабораторна робота №2

з дисципліни «Введення до операційних систем»

«Синхронізація процесів»

Виконав студент групи: КВ-11

ПІБ: Терентьєв Іван Дмитрович

Перевірив: _____

Київ 2024

Загальне завдання

1. Розробити програму, що моделює роботу заданого об'єкта, використовуючи для доступу процесів до подільних ресурсів засоби синхронізації. Пристрій, що моделюється, і засоби синхронізації процесів визначаються варіантом завдання. Вхідні дані студент задає самостійно з урахуванням особливостей індивідуального завдання.
2. Забезпечити візуалізацію роботи моделі з наглядною демонстрацією результатів.

3. Проаналізувати та пояснити отримані результати. За результатами роботи надати висновки щодо використаних засобів синхронізації.

Індивідуальне завдання за варіантом 23(8)

Об'єкт моделювання:

Автомат для продажу авіа білетів. Автомат приймає гроші (тут тільки одного визначеного номіналу – 1грн.) і видає здачу монетами вартістю до 1 грн. (1, 2, 5, 10, 25, 50 коп.). Сума здачі розраховується. Початкова кількість монет кожного номіналу задається і становить: 1 коп. – 50 шт., 2 коп. – 25 шт.,
5 коп. – 20 шт., 10 коп. – 15 шт.,
25 коп. – 10 шт., 50 коп. – 5 шт.,
Введення запиту на продаж здійснюється шляхом вибору певного пункту меню:

0 – включити автомат, 1 – купити білет до Києва вартістю 28 коп., 2 – білет до Москви вартістю 37 коп., 3 – білет до Лондона вартістю 50 коп., 4 – білет до Берлина вартістю 77 коп., 5 – білет до Парижа вартістю 91 коп.

Якщо здачу видати можливо, програма формує потрібний набір монет для здачі (також коригує банк монет) і формує сигнал на видачу. Якщо потрібних купюр для здачі не достає, формується відповідне повідомлення. Вимоги на видачу грошей надходять після чергового сеансу продажу або відмови.

Кількість терміналів і процесів:

Модель автомата представити у вигляді двох взаємодіючих процесів A і B. Процес A визначає факти надходження вимог на продаж і потрібну суму здачі. Процес B очікує момент появи необхідності видати здачу і, якщо величина здачі відома, визначає кількість і номінали потрібних монет або неможливість видачі.

Засоби синхронізації:

Для організації доступу до подільних ресурсів використати семафори.

Код програми:

main.c

```
#include "coin_machine.h"
#include "sem_blocks.h"
#include "thread_A.h"
#include "thread_B.h"
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define ERROR_CREATE_THREAD -11
#define SUCCESS 0
#define TIMES_TO_GO 25
// #define GET_CHAR_MODE

/*
Kiev 28
Moscow 37
London 50
Berlin 77
Paris 91
*/

int main() {
  CM coin_mch1 = {
       .change = 0,
       .COINS_IN_MACHINE =
           {{1, 50}, {2, 25}, {5, 20}, {10, 15}, {25, 10}, {50, 5}},
       .COINS_TO_CHANGE = {{1, 0}, {2, 0}, {5, 0}, {10, 0}, {25, 0}, {50, 0}},
       .PRICES = {28, 37, 55, 77, 91},
       .message = ""};
  srand((unsigned int)time(NULL));
  pthread_t PA, PB;
  int status1, status2;
  int status_addr1, status_addr2;
#ifdef GET_CHAR_MODE
  size_t i = 0;
  while (1) {
#endif
#ifndef GET_CHAR_MODE
    for (size_t i = 0; i < TIMES_TO_GO; i++) {
#endif
      printf("Today tickets bought: %ld\n", i);
      sem_init(&sem1, 0, 1);
      sem_init(&sem2, 0, 0);
      sem_init(&sem3, 0, 0);
      status1 = pthread_create(&PA, NULL, thread_a, (void *)&coin_mch1);
      if (status1 != 0) {
        printf("int main() error: cannot create thread_a");
        exit(ERROR_CREATE_THREAD);
      }
      status2 = pthread_create(&PB, NULL, thread_b, (void *)&coin_mch1);
      if (status2 != 0) {
        printf("int main() error: cannot create thread_b");
        exit(ERROR_CREATE_THREAD);
      }

      status1 = pthread_join(PA, (void **)&status_addr1);
      status2 = pthread_join(PB, (void **)&status_addr2);
#ifdef GET_CHAR_MODE
```

```
      getchar();
      i++;
#endif
    }
    pthread_exit(NULL);
  }
```

coin_machine.h

```
#ifndef COIN_MACHINE_H
#define COIN_MACHINE_H
struct coin_machine {
  unsigned long int change;
  unsigned long int COINS_IN_MACHINE[6][2];
  unsigned long int COINS_TO_CHANGE[6][2];
  unsigned long int PRICES[5];
  char *message;
};
typedef struct coin_machine CM;
#endif
```

sem_blocks.c

```
#include "sem_blocks.h"
sem_t sem1;
sem_t sem2;
sem_t sem3;
```

sem_blocks.h

```
#ifndef SEM_BLOCKS_H
#define SEM_BLOCKS_H
#include <semaphore.h>
extern sem_t sem1;
extern sem_t sem2;
extern sem_t sem3;
#endif
```

thread_A.c

```
#include "thread_A.h"
#include "coin_machine.h"
#include "sem_blocks.h"
#include <semaphore.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

void calculate_change(CM *c_machine) {
  unsigned long int client_gives = 100;
  int selected_ticket = rand() % 5;
  char *cities[5] = {"Kiev", "Moscow", "London", "Berlin", "Paris"};
  printf("I want to buy ticket to %s\n", cities[selected_ticket]);
  c_machine->change = client_gives - c_machine->PRICES[selected_ticket];
}
void give_change(CM *c_machine) {

  printf("%s\n", c_machine->message);
  printf("Change: %ld\n", c_machine->change);
```

```
  printf("Coins to give:\n");
  for (size_t i = 0; i < 6; i++) {
    printf("[%ld|%ld] ", c_machine->COINS_TO_CHANGE[i][0],
           c_machine->COINS_TO_CHANGE[i][1]);
  }
  printf("\nCoins in machine:\n");
  for (size_t i = 0; i < 6; i++) {
    printf("[%ld|%ld] ", c_machine->COINS_IN_MACHINE[i][0],
           c_machine->COINS_IN_MACHINE[i][1]);
  }
  printf("\n");
  c_machine->change = 0;
  if (strcmp(c_machine->message, "Success") == 0) {
    for (size_t i = 0; i < 6; i++) {
      c_machine->COINS_IN_MACHINE[i][1] -= c_machine->COINS_TO_CHANGE[i][1];
    }
  }

  CM temp_machine = {
      .COINS_TO_CHANGE = {{1, 0}, {2, 0}, {5, 0}, {10, 0}, {25, 0}, {50,
0}}};

  for (int i = 0; i < 6; i++)
    for (int k = 0; k < 2; k++)
      c_machine->COINS_TO_CHANGE[i][k] = temp_machine.COINS_TO_CHANGE[i][k];
}
void *thread_a(void *args) {
  sem_wait(&sem1);
  printf("==================================================\n");
  CM *c_machine = (CM *)args;
  calculate_change(c_machine);
  sem_post(&sem2);
  sem_wait(&sem3);
  give_change(c_machine);
  printf("==================================================\n");
  sem_post(&sem1);
  return NULL;
}
```

thread_A.h

```
#ifndef THREAD_A_H
#define THREAD_A_H
void *thread_a(void *args);
#endif
```

thread_B.c

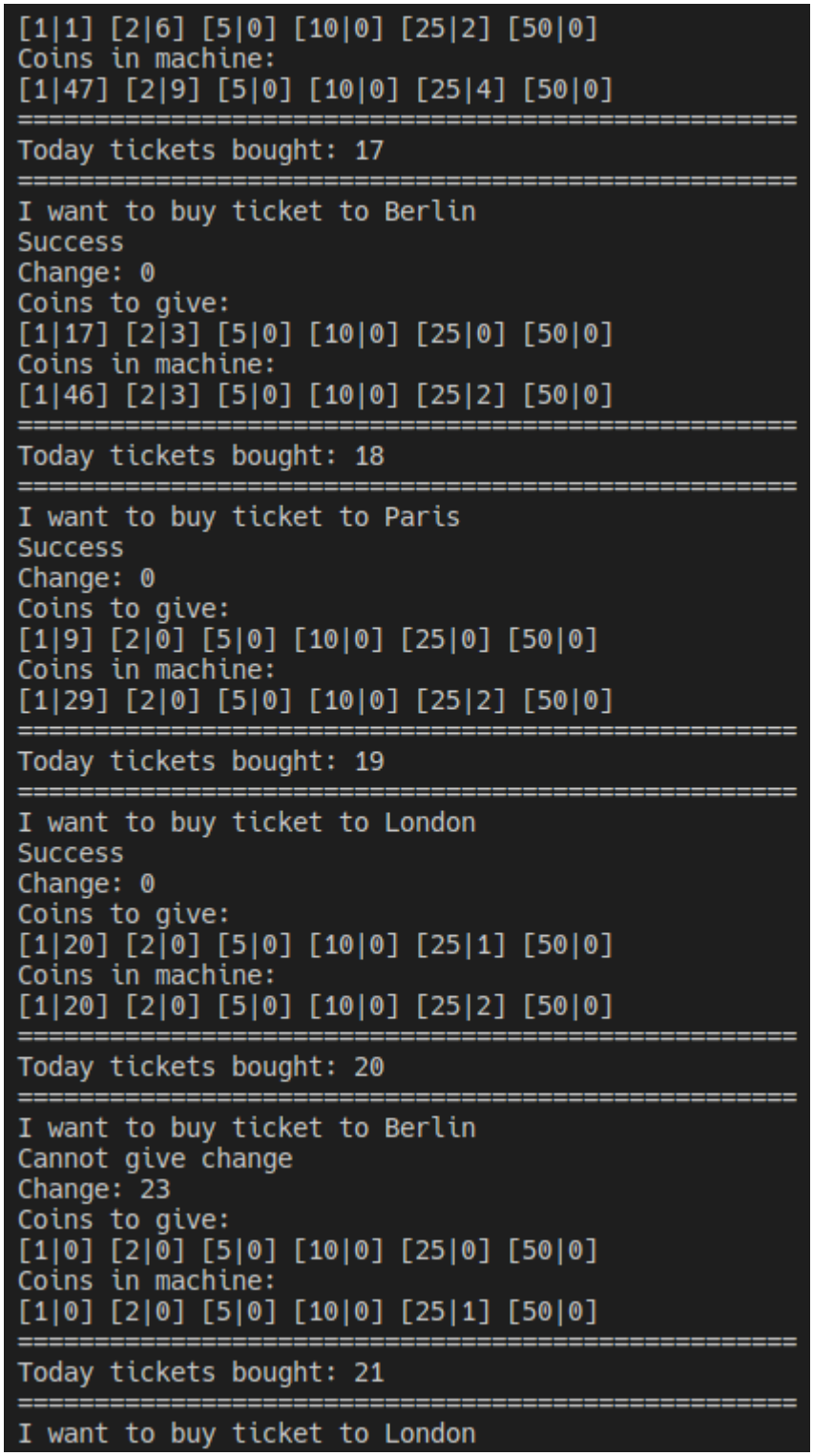
```
#include "thread_B.h"
#include "coin_machine.h"
#include "sem_blocks.h"
#include <semaphore.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

void calculate_coins(CM *c_machine) {
  int i = 5;
  while (i >= 0) {
    if ((c_machine->COINS_IN_MACHINE[i][1] -
         c_machine->COINS_TO_CHANGE[i][1]) != 0) {
      if (c_machine->change > c_machine->COINS_IN_MACHINE[i][0]) {
```

```
        c_machine->change -= c_machine->COINS_IN_MACHINE[i][0];
        c_machine->COINS_TO_CHANGE[i][1]++;
      } else if (c_machine->change == c_machine->COINS_IN_MACHINE[i][0]) {
        c_machine->change -= c_machine->COINS_IN_MACHINE[i][0];
        c_machine->COINS_TO_CHANGE[i][1]++;
        break;
      } else {
        i--;
      }
    } else {
      i--;
    }
  }
}
void generate_change(CM *c_machine) {
  if (c_machine->change != 0)
    c_machine->message = "Cannot give change";
  else
    c_machine->message = "Success";
}
void *thread_b(void *args) {
  sem_wait(&sem2);
  CM *c_machine = (CM *)args;
  calculate_coins(c_machine);
  generate_change(c_machine);
  sem_post(&sem3);
  return NULL;
}
```

thread_B.h

```
#ifndef THREAD_B_H
#define THREAD_B_H
void *thread_b(void *args);
#endif
```

Скріншот програми:



Висновок:

Під час виконання лабораторної роботи була розроблена програма, що моделювала роботу заданого об’єкта, а саме автомату для продажу авіа білетів, використовуючи для доступу процесів до подільних ресурсів засоби синхронізації, а саме семафори. Була забезпечена візуалізація роботи моделі з наглядною демонстрацією результатів. В результаті роботи програми можна побачити, що семафори забезпечили безпечний доступ до спільних ресурсів між двома потоками A і B. Де спочатку A очікує купівлю білету, а далі розраховує решту, B отримує дозвіл на розрахунок потрібних монет які будуть видані як решта та генерує повідомлення якщо видати решту неможливо(в автоматі не вистачає потрібних номіналів монет), та повертаючись до A виводяться відповідні повідомлення та якщо можливо, видається решта. Авжеж автомат намагається завжди видати решту найбільшими номіналами монет, тому в якийсь момент, коли монет й меншого номіналу починає не вистачати, автомат перестає видавати решту та продавати квитки.