

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 А і В. Процес А визначає факти надходження вимог на продаж і потрібну суму здачі. Процес В 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*

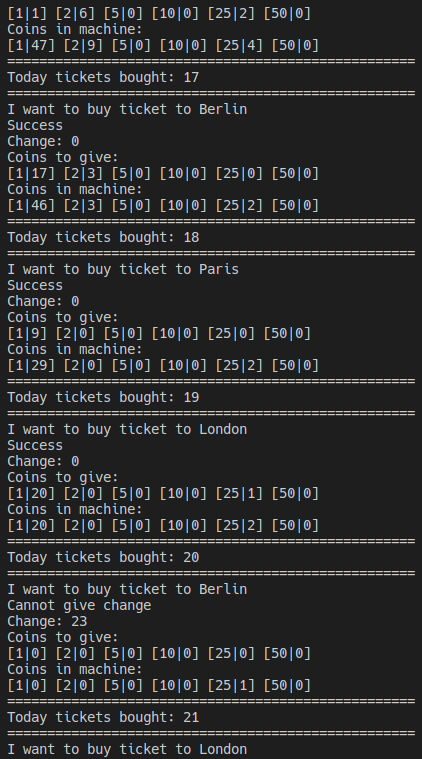
#ifndef THREAD\_B\_H

#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 А і B. Де спочатку А 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 А 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.