

**Лабораторна робота №2**

З дисципліни

**Введеня до операційних систем**

**Тема: “Синхронізація процесів”**

Виконав: студент 3 курсу

ФПМ групи КВ-91

Зиков А.С.

Перевірила: Дробязко І.П.

Київ - 2022р.

**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.

**Варіант 13**

***Об’єкт моделювання: Банкомат.***

Банкомат приймає заявку на виплату грошей на суму X, порівнює її з кількістю грошей на рахунку клієнта Y і, якщо X<Y, здійснює спробу виплати певними купюрами номіналом 1, 2, 5, 10, 20, 50, 100 грн. Величини X і Y – випадкові числа. Початкова кількість купюр різних номіналів у банкоматі задається студентом. Якщо сума X підлягає видачі, програма визначає, чи можна цю суму виплатити наявними у даний момент купюрами. Якщо так – гроші видаються (також коригується кількість наявних у банкоматі купюр відповідного номіналу), якщо ні – банкомат повідомляє про відмову. Вимоги на видачу грошей надходять до банкомату після чергового обслуговування клієнта (або відмови).

***Кількість терміналів і процесів:*** Модель автомата представити у вигляді двох взаємодіючих процесів *А і В*.

Процес *А* визначає факт надходження вимоги на виплату та можливість виплати потрібної суми, фіксує можливість виплати. Процес *В* очікує момент появи необхідності зробити виплату і, якщо виплата можлива, визначається необхідна кількість купюр кожного номіналу для виплати, коригується рахунок клієнта і кількість готівки у банкоматі, йде виплата. Якщо виплата неможлива – клієнту надається про це повідомлення.

***Засоби синхронізації:*** Для організації доступу до подільних ресурсів використати поштові скриньки*.*

**Лістинг програми**

(код також викладений на моєму GitHub: <https://github.com/Luritron/OS>)

**Struct.h**

#pragma once

#include<iostream>

#include<thread>

#include<functional>

#include<mutex>

#include<condition\_variable>

struct PostBox

{

bool transaction\_demand = false;

bool is\_transaction\_possible = false;

std::condition\_variable g\_queuecheck;

};

struct Cell

{

int wallet = 100;

int one = 50;

int two = 25;

int five = 20;

int ten = 15;

int twenty = 10;

int fifty = 5;

int hundred = 5;

void cash(int value)

{

//printf("You get : %d \n ", value);

wallet += value;

while (value != 0)

{

if (value >= 100)

{

value -= 100;

hundred -= 1;

}

else if (value >= 50)

{

value -= 50;

fifty -= 1;

}

else if (value >= 20)

{

value -= 20;

twenty -= 1;

}

else if (value >= 10)

{

value -= 10;

ten -= 1;

}

else if (value >= 5)

{

value -= 5;

five -= 1;

}

else if (value >= 2)

{

value -= 2;

two -= 1;

}

else if (value >= 1)

{

value -= 1;

one -= 1;

}

}

}

};

**Main.cpp**

#include "Struct.h"

void Thread1(int& sum, Cell& cell, PostBox& port);

void Thread2(int& sum, Cell& cell, PostBox& port);

std::mutex mutex\_thread;

bool transaction;

int main()

{

Cell cell;

PostBox port;

int sum = 0;

std::thread thread1(Thread1, std::ref(sum), std::ref(cell), std::ref(port));

std::thread thread2(Thread2, std::ref(sum), std::ref(cell), std::ref(port));

thread1.join();

thread2.join();

std::cout << "\nYour wallet balance now: " << cell.wallet << std::endl;

std::cout << "\nRemaining banknotes:" << std::endl;

std::cout << "\"Hundred\" currency unit - "<<cell.hundred<<std::endl;

std::cout << "\"Fifty\" currency unit - "<<cell.fifty<<std::endl;

std::cout << "\"Twenty\" currency unit - "<<cell.twenty<<std::endl;

std::cout << "\"Ten\" currency unit - "<<cell.ten<<std::endl;

std::cout << "\"Five\" currency unit - "<<cell.five<<std::endl;

std::cout << "\"Two\" currency unit - "<<cell.two<<std::endl;

std::cout << "\"One\" currency unit - "<<cell.one<<std::endl;

return 0;

}

void Thread1(int& sum, Cell& cell, PostBox& port) // Signalizing thread A

{

std::unique\_lock<std::mutex> lk(mutex\_thread);

//printf("Thread1 started \n "); // debug solution

std::cout << "ATM has: " << cell.wallet << std::endl;

std::cout << "Choose sum: ";

std::cin >> sum;

port.transaction\_demand = true;

if (sum > cell.wallet)

{

port.is\_transaction\_possible = false;

}

else

{

port.is\_transaction\_possible = true;

}

transaction = true;

lk.unlock();

port.g\_queuecheck.notify\_one(); // Opens thread2

}

void Thread2(int& sum, Cell& cell, PostBox& port) // Executing thread B

{

//printf("Thread2 started \n\n "); // debug solution

std::unique\_lock<std::mutex> lk(mutex\_thread);

port.g\_queuecheck.wait(lk, [] {return transaction; });

if (port.is\_transaction\_possible == false)

{

std::cout << "Your request higher than the available amount" << std::endl;

std::terminate();

}

else if (port.is\_transaction\_possible == true && port.transaction\_demand == true)

{

cell.cash(sum);

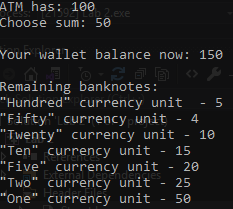
}

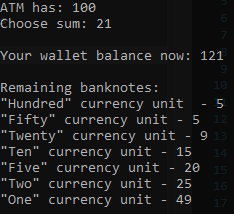
lk.unlock();

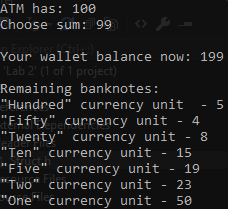
port.g\_queuecheck.notify\_one();

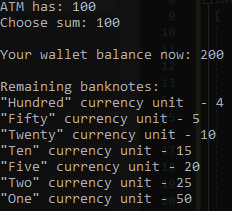
}

**Тестування**









VsDebugConsole_Zw0BpIGZxz

**Висновок**

При виконанні даної лабораторної роботи використано синхронізацію 2 потоків згідно варіанту завдання. Спільний ресурс складається з атомарних змінних, доступ до критичнох секції в потокових функціях здійснюється завдяки принципу “поштової скриньки”. Цей елементарний прийом синхронізації досить зручно використовувати для синхронізації двох потоків A та В, як у нашому випадку, оскільки, на відмінну від семафору, для процесу синхронізаці потоків ми використовуємо не процес зміни значення атомарної змінної, буфер в якому знаходяться повідомлення. Також, для синхронізації потоків, використовуються “гнізда” - функції для обробки прийнятих повідомлень, як це реалізовано в даній лабораторній роботі. По завершеню операції перевірки на допустимість зняття суми, що не перебільшує наявний номінал, перший потік надсилатиме одне з повідомлень, що свідчить про можливість перейти до роботи наступному потоку, що очікує на сигнал у вигляді функції-сокету. У разі, якщо перший потік передав інформацію про те, що він закінчив власні операції та надав повідомлення про можливість знаття коштів, в такому випадку в потоці здійснюється операція обрахунку введеної користувачем суми. Таким чином, завдяки реалізованому буферу із повідомленнями, потоки мають можливість спілкуватися між собою та виконувати послідовно поставлені задачі.