#include <iostream>

#include <string>

#define HAVE\_STRUCT\_TIMESPEC

#include <pthread.h>

#include <thread> // для this\_thread::sleep\_for

#include <semaphore.h>

#include <stdlib.h>

#include <stdio.h>

#include <time.h>

enum class Components

{

tobacco,

paper,

matches

};

class Smoker

{

private:

Components \_comp;

std::string \_name;

public:

Smoker(const Components& component, const std::string& name)

{

\_comp = component;

\_name = name;

}

~Smoker() {}

Components GetComponent() const

{

return \_comp;

}

void Smoke() const

{

std::cout << \_name << " курильщик - пых пых" << '\n';

}

};

sem\_t empty;

sem\_t full;

pthread\_mutex\_t mutexW; //мютекс для операции записи

pthread\_mutex\_t mutexR; //мютекс для операции чтения

Components not\_exist = static\_cast<Components>(rand() % 3);

void\* Producer(void\* param)

{

while (true)

{

std::this\_thread::sleep\_for(std::chrono::milliseconds(2000));

//защита на запись

pthread\_mutex\_lock(&mutexW);

sem\_wait(&empty);

std::cout << "\nproduce\n";

// отсутствующий компонент, который есть у потребителя

not\_exist = static\_cast<Components>(rand() % 3);

//количество возможных потребителей = 3

sem\_post(&full);

sem\_post(&full);

sem\_post(&full);

pthread\_mutex\_unlock(&mutexW);

}

return nullptr;

}

void\* Consumer(void\* param)

{

// засыпаем сначала, чтоб производитель заработал

std::this\_thread::sleep\_for(std::chrono::milliseconds(500));

Smoker& smoker = \*((Smoker\*)param);

while (true)

{

// защита на чтение

pthread\_mutex\_lock(&mutexR);

sem\_wait(&full);

std::cout << "consume ";

if (smoker.GetComponent() == not\_exist)

{

std::cout << '\n';

smoker.Smoke();

// обновляем продукцию

sem\_post(&empty);

}

pthread\_mutex\_unlock(&mutexR);

std::this\_thread::sleep\_for(std::chrono::milliseconds(1000));

}

}

int main()

{

srand(time(NULL));

setlocale(LC\_ALL, "Russian");

pthread\_mutex\_init(&mutexW, nullptr);

pthread\_mutex\_init(&mutexR, nullptr);

sem\_init(&empty, 0, 1); //количество свободных ячеек = 1

sem\_init(&full, 0, 0); //количество занятых ячеек = 0

// создаем и запускаем производителя

pthread\_t thread\_producer;

pthread\_create(&thread\_producer, nullptr, Producer, nullptr);

// создаем и запускаем потребителей

pthread\_t threads[3];

Smoker\* smoker\_1 = new Smoker(Components::tobacco, "1");

Smoker\* smoker\_2 = new Smoker(Components::paper, "2");

Smoker\* smoker\_3 = new Smoker(Components::matches, "3");

pthread\_create(&threads[0], nullptr, Consumer, (void\*)(smoker\_1));

pthread\_create(&threads[1], nullptr, Consumer, (void\*)(smoker\_2));

pthread\_create(&threads[2], nullptr, Consumer, (void\*)(smoker\_3));

// зацикливаем главный поток, дожидаясь производителя

pthread\_join(thread\_producer, nullptr);

}