НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Дисциплина: «Архитектура вычислительных систем»

Задача о читателях и писателях-3 («подтвержденное чтение»)

Пояснительная записка

Вариант 9

Выполнил: Гарифуллин Руслан Ильфатович

студент БПИ191,

ФКН Программная инженерия

Москва 2020

1. Текст задания

Базу данных разделяют два типа процессов – читатели и писатели. Читатели выполняют транзакции, которые просматривают записи базы данных, транзакции писателей и просматривают и изменяют записи. Предполагается, что в начале БД находится в непротиворечивом состоянии (т.е. отношения между данными имеют смысл). Каждая отдельная транзакция переводит БД из одного непротиворечивого состояния в другое. Транзакции выполняются в режиме «подтвержденного чтения», то есть процесс-писатель не может получить доступ к БД в том случае, если ее занял другой процесс-писатель или процесс-читатель. К БД может обратиться одновременно сколько угодно процессов-читателей. Процесс читатель получает доступ к БД, даже если еезанял процесс-писатель. Создать многопоточное приложение с потоками-писателями и потоками-читателями. Реализовать решение, используя **семафоры**, и **не используя блокировки чтения-записи**.

Программа должна быть выполнения на языке C++ и быть скомпилирована при помощи кроссплатформенного компилятора GCC C++.

2. Описание расчетных методов

Программа использует POSIX threads для распараллеливания работы читателей и писателей. Используется в сумме три семафора:

- **writing**, блокировка процесса записи между потоками записи;
- **write_access**, блокировка процесса записи при занятии потоками чтения и записи;
- **cout_access**, семафор для последовательного вывода информации в консоль.

Программа представляет собой некое подобие лотереи, где игроки регистрируются (*Writer*) на получение выигрыша в буфер из **SIZE=100** человек, из которых случайным образом избираются победители (*Reader*), которым выдается выигрыш, зависящий от коэффицента везения человека.

3. Описание входных данных

Скомпилированная программа представляет собой бинарный файл **main**, при запуске которого возможно указание количества писателей и читателей при помощи аргументов командной строки:

- **./main 4 7** – количество писателей: 4, читателей: 7;
- **./main 5 –** количество писателей: 5, читателей: 5;
- **./main –** значение по умолчанию: 5 писаталей, 3 читателя.

4. Описание выходных данных

Вывод программы представляет собой текст в консоли, где в каждой строке описывается о проведенных действиях: записи игрока в базу данных, выдачи выигрыша, состоянии потоков:

```
Player registrar 1 started.
Lottery 0 started.
Player registrar 0 started.
Lottery 1 started.
Lottery 1 gave out $29.26 to the player udgFpkWvVf, record #6.
Lottery 0 gave out $46.50 to the player juTCecCbRu, record #2.
Registrar 1 placed a new player pahDGgHgwm into record #9.
Registrar 0 placed a new player hOIkhgTmUc into record #6.
Lottery 1 gave out $46.50 to the player juTCecCbRu, record #2.
Lottery 0 gave out $45.96 to the player mnvZTquoxP, record #5.
```

Пример выходных данных

5. Описание ключевых переменных

Игроки лотереи хранятся в массиве **db** в виде **struct**-ов со следующей структурой:

- **name**, имя игрока (случайно генерируемая строка);
- **age**, целочисленный возраст игрока [18; 99];
- **luck**, коэффицент удачи, влияет на размер выигрыша.

6. Дополнительные библиотеки

Были использованы функции и структуры данных из стандартной библиотеки C++ (string, iostream), а также механизмы работы с семафорами <semaphore.h> и библиотека POSIX threads <pthread.h>.

Приложение 1

Код программы

Код программы представлен в репозитории на сервисе GitHub: https://github.com/ruslang02/HSE-FASM-Projects/tree/master/project02.

```
#include <iostream>
#include <iomanip>
#include <string>
#include <pthread.h>
#include <semaphore.h>
#include <unistd.h>
#define SIZE 100

int randAdd = 0;
// Generates a random string (player's name).
std::string random_str()
{
  std::string tmp_s;
  static const char alpha[] =
"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz";
  srand((unsigned)time(NULL) * getpid() + randAdd);
  tmp_s.reserve(10);
  for (int i = 0; i < 10; ++i)
    tmp_s += alpha[rand() % (sizeof(alpha) - 1)];
  randAdd++;
  return tmp_s;
}

// Stores lottery player's info, autogenerates its properties.
struct LotteryPlayer
{
  std::string name;
  int age;
  double luck;
  LotteryPlayer()
  {
    name = random_str();
    age = (random() % 82) + 18;
    luck = static_cast<float>(rand()) / static_cast<float>(RAND_MAX);
  }

  ~LotteryPlayer() {
    name.~basic_string();
  }
};
// A "database" for the players.
LotteryPlayer *db[SIZE];
```

```
sem_t writing, write_access, cout_access;
// Writing thread which registers players and places them in DB.
void *Writer(void *args)
{
  int thread_num = *((int *)args);
  sem_wait(&cout_access);
  std::cout << "Player registrar " << thread_num << " started.\n";
  sem_post(&cout_access);

  while (1)
  {
    int index = random() % SIZE;
    sem_wait(&write_access);
    sem_wait(&writing);
    delete db[index];
    LotteryPlayer *player = db[index] = new LotteryPlayer();
    sem_wait(&cout_access);
    std::cout << "Registrar " << thread_num << " placed a new player " << player-
>name << " into record #" << index << ".\n";
    sem_post(&cout_access);
    sem_post(&writing);
    sem_post(&write_access);
    sleep(random() % 3 + 1);
  }
  return nullptr;
}
// Reader thread, gives out money to people.
void *Reader(void *args)
{
  int thread_num = *((int *)args);
  sem_wait(&cout_access);
  std::cout << "Lottery " << thread_num << " started.\n";
  sem_post(&cout_access);

  while (1)
  {
    int index = random() % SIZE;
    sem_trywait(&write_access);
    LotteryPlayer *player = db[index];
    double money = player->luck * (((random() / RAND_MAX) % 450) + 50);
    sem_wait(&cout_access);
    std::cout << std::fixed << std::setprecision(2) <<
    "Lottery " << thread_num << " gave out $" << money <<
    " to the player " << player->name << ", record #" << index << ".\n";
    sem_post(&cout_access);
    int w_val;
    sem_getvalue(&write_access, &w_val);
```

```
    if (w_val < 1) sem_post(&write_access);
    sleep(1);
  }
  return nullptr;
}
int main(int argc, char** argv)
{
  srand (static_cast <unsigned> (time(0)));
  int WRITER_COUNT = 5, READER_COUNT = 3;
  switch(argc) {
    case 2:
      WRITER_COUNT = READER_COUNT = atoi(argv[1]);
      break;
    case 3:
      WRITER_COUNT = atoi(argv[1]);
      READER_COUNT = atoi(argv[2]);
      break;
  }
  sem_init(&writing, 0, 1);
  sem_init(&write_access, 0, 1);
  sem_init(&cout_access, 0, 1);
  for (int i = 0; i < SIZE; i++)
  {
    LotteryPlayer *player = db[i] = new LotteryPlayer();
    std::cout << std::setprecision(2) << "Registered player " << player->name << ",
age " << player->age << " with luck ratio of " << player->luck << ".\n";
  }

  std::cout << "Welcome to the big lottery! We have a 100-player buffer, from which
we select our winners, so you need to register yourself each time your place gets
occupied by someone else. Good luck.\n";
  pthread_t writer[WRITER_COUNT], reader[READER_COUNT];
  int writers[WRITER_COUNT], readers[READER_COUNT];
  for (int i = 0; i < WRITER_COUNT - 1; i++)
  {
    writers[i] = i + 1;
    pthread_create(&writer[i], nullptr, Writer, (void *)(writers + i));
  }
  for (int i = 0; i < READER_COUNT; i++)
  {
    readers[i] = i;
    pthread_create(&reader[i], nullptr, Reader, (void *)(readers + i));
  }
  int i = 0;
  Writer((void *)&i);
  return 0;
}
```

Список использованных источников

- http://softcraft.ru/edu/comparch/lect/07-parthread/multitreading.pdf
- https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D0%BC%D0%B0%D1%84%D0%BE%D1%80_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)
- https://docs.oracle.com/cd/E19455-01/806-5257/sync-73219/index.html