Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет информационных технологий и управления

Кафедра вычислительных методов и программирования

Реферат
по дисциплине «Основы информационных технологий»
на тему «Создание многопоточных приложений»

Выполнил магистрант группы 6М1911 Будный Р. И. _____

Проверил д.ф.-м.н. Колосов С. В. _____

Минск 2016

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Современные операционные системы предоставляют средства для многопоточного программирования. Каждая исполняемая программа в рамках своего процесса может иметь один или несколько потоков выполнения. Поток выполнения — наименьшая единица обработки задачи, исполнение которой может быть назначено ядром операционной системы [1]. Операционная система выделяет каждому потоку некоторый короткий промежуток времени, в течение которого происходит выполнение активного потока. По истечении этого промежутка активный поток блокируется, и операционная система переходит к следующему. Таким образом достигается видимость одновременного выполнения нескольких потоков в рамках одного процесса — многопоточность.

Основным отличием потоков выполнения от процессов является использование общего адресного пространства. Потоки в рамках одного процесса выполнения имеют общий доступ к данным и другим ресурсам, предоставляемых данному процессу операционной системой.

Существуют различные программные реализации многопоточности. В различных ОС потоки могут быть реализованы на уровне ядра, в пользовательском пространстве, или с использованием различных гибридных схем. Каждый из этих подходов имеет ряд достоинств и недостатков.

Программный интерфейс управления потоками также зависит от конкретной операционной системы. Наряду с этим, существуют стандартизированные программные интерфейсы управления потоками. Наиболее известным из них является стандарт управления потоками POSIX. В данном реферате производится обзор данного стандарта, на его примере рассматриваются основные операции управления потоками, а также базовые примитивы синхронизации. Кроме этого, рассматриваются средства поддержки многопоточности, предоставляемые стандартной библиотекой языка C++.

1 СТАНДАРТ РЕАЛИЗАЦИИ ПОТОКОВ ВЫПОЛНЕНИЯ POSIX

В прошлом программные интерфейсы управления потоками, предоставляемые различными операционными системами, существенно различались. Этот факт значительно усложнял написание кроссплатформенных программ. В 1995 году был разработан и опубликован стандарт программного интерфейса потоков IEEE POSIX 1003.c. Он представляет собой набор связанных типов, функций и констант языка C, позволяющих управлять жизненным циклом потоков и выполнять их синхронизацию. На данный момент программные интерфейсы упрвления потоками, предоставляемые практически всеми операционными системами на базе UNIX, являются POSIX-совместимыми [2].

В соответствии со стандартом, программный интерфейс управления потоками POSIX (Pthreads) описан в заголовочном файле `<pthreads.h>`. Следует отметить, что данный заголовочный файл не является частью стандартной библиотеки языка C.

1.1 Управление жизненным циклом потока

Рассмотрим основные функции и типы данных, определенные стандартом Pthreads и предназначенные для управления потоками.

Функция `pthread_create()` предназначена для создания и запуска нового потока. Она принимает следующие аргументы:

– `pthread_t* thread` — идентификатор потока;

– `const pthread_attr_t* attr` — атрибуты потока;

– `void *(*start_routine)(void*)` — функция, предназначенная для выполнения в новом потоке;

– `void* arg` — аргумент, передаваемый в `start_routine`.

Данная функция возвращает нулевое значение в случае успеха или код ошибки в противном случае.

Функция `pthread_exit()` предназначена для завершения вызывающего потока. Он имеет параметр `void* value_ptr`, предназначенный для передачи возвращаемого значения в поток, ожидающий завершения.

Функции pthread_attr_init и pthread_attr_destroy предназначены для инициализации и удаления структуры атрибутов потока соответственно. На рисунке 1 представлен простейший пример создания потока.

```
struct thread_data{
    int thread_id;
    int sum;
    char* message;
};

struct thread_data thread_data_array[NUM_THREADS];

void *printHello(void* threadarg) {
    struct thread_data* my_data;
    ...
    my_data = (struct thread_data *) threadarg;
    taskid = my_data->thread_id;
    sum = my_data->sum;
    hello_msg = my_data->message;
    ...
}

int main(int argc, char** argv) {
    ...
    thread_data_array[t].thread_id = t;
    thread_data_array[t].sum = sum;
    thread_data_array[t].message = messages[t];
    rc = pthread_create(
        &threads[t], NULL, printHello,
        (void *) &thread_data_array[t]);
    ...
}
```

Рисунок 1 – Создание потока POSIX

Здесь в функции main() выполняется создание потока, выполняющего функцию printHello() с использованием аргумента, представляющего собой структуру thread_data. Следует отметить, что в тех случаях, когда исполнение функции потока заканчивается, выход из потока выполняется автоматически.

2 СРЕДСТВА СТАНДАРТНОЙ БИБЛИОТЕКИ ЯЗЫКА C++ ДЛЯ ОРГАНИЗАЦИИ МНОГОПОТОЧНЫХ ВЫЧИСЛЕНИЙ

*** Управление жизненным циклом потока + std::thread + std::mutex + std::lock_guard + std::condition_varibale + std::promise + std::future + std::async
*** Примитивы синхронизации

ЗАКЛЮЧЕНИЕ

...

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Поток выполнения [Электронный ресурс]. — https://en.wikipedia.org/wiki/Thread_(computing) : [б. и.].

[2] POSIX Threads Programming [Электронный ресурс]. — [Б. м. : б. и.]. — Режим доступа: `https://computing.llnl.gov/tutorials/pthreads/`.