**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра информационных систем**

отчет

**по лабораторной работе №1**

**по дисциплине «Операционные системы»**

Тема: «**СОЗДАНИЕ И УНИЧТОЖЕНИЕ ПОТОКОВ»**

|  |  |  |
| --- | --- | --- |
| Студент гр. 8392 |  | Артемов И.Р. |
| Преподаватель |  | Широков В.В. |

Санкт-Петербург

2020

**Цель работы**

Знакомство с базовой структурой построения многопоточной программы и с системными вызовами, обеспечивающими создание и завершение потоков.

**Основные теоретические положения**

Базовая структура многопоточной программы, взятая за основу всех работ, выглядит следующим образом:

1. Описываются поточные функции, соответствующие потокам программы.
2. В основной программе создаются потоки на основе поточных функций.
3. После создания потоков основная программа приостанавливает выполнение и ожидает команды завершения.
4. При поступлении команды завершения основная программа формирует команды завершения потоков.
5. Основная программа переходит к ожиданию завершения потоков.
6. После завершения потоков основная программа завершает свою работу.

**Экспериментальные результаты**

Вывод из консоли:

ilya@ilya-VirtualBox:~/Документы/pr/pr\_labs/labs/lab1$ ./q

first thread started

1second thread started

212122121212121212121122112211221122112213122112122112

first thread ended with code : ok

second thread ended with code : ok

Main ended

Как видно из результата выполнения программы потоки работают асинхронно – данные выводятся в случайном порядке. После выставления флага завершения прекращает работу.

**Ответы на вопросы**

*1. В чем состоит различие между понятиями «поток» и «процесс»?*

Процесс — экземпляр программы во время выполнения, независимый объект, которому выделены системные ресурсы (например, процессорное время и память). Каждый процесс выполняется в отдельном адресном пространстве: один процесс не может получить доступ к переменным и структурам данных другого. Если процесс хочет получить доступ к чужим ресурсам, необходимо использовать межпроцессное взаимодействие. Это могут быть конвейеры, файлы, каналы связи между компьютерами и многое другое.

Поток использует то же самое пространства стека, что и процесс, а множество потоков совместно используют данные своих состояний. Как правило, каждый поток может работать (читать и писать) с одной и той же областью памяти, в отличие от процессов, которые не могут просто так получить доступ к памяти другого процесса. У каждого потока есть собственные регистры и собственный стек, но другие потоки могут их использовать.

Поток — определенный способ выполнения процесса. Когда один поток изменяет ресурс процесса, это изменение сразу же становится видно другим потокам этого процесса.

*2. Как осуществить передачу параметров в функцию потока при создании потока?*

При выполнении функции создания потока в нее можно передать аргумент

*3. Какие способы завершения потока существуют?*

* Вызовом оператора *return* из функции потока;
* Вызовом функции: *int pthread\_cancel(pthread\_t thread)* из другого потока;
* Вызовом функции: *int pthread\_exit(void \*value\_ptr).*

*4. На какие характеристики потока можно влиять через атрибуты потока?*

|  |  |  |
| --- | --- | --- |
| Атрибут | Значение default | Смысл |
| scope | PTHREAD\_SCOPE\_PROCESS | Новый поток не ограничен - не присоединен ни к одному процессу |
| detachstate | PTHREAD\_CREATE\_JOINABLE | Статус выхода и поток сохраняются после завершения потока |
| stackaddr | NULL | Новый поток получает адрес стека, выделенного системой |
| stacksize | 1 Мбайт | Новый поток имеет размер стека, определенный системой |
| inheritsched | PTHREAD\_INHERIT\_SCHED | Поток наследует приоритет диспетчеризации родительского потока |
| schedpolicy | SCHED\_OTHER | Новый поток использует диспетчеризацию с фиксированными приоритетами. Поток работает, пока не будет прерван потоком с высшим приоритетом или не приостановится |

*5. В каких состояниях может находиться поток?*

Может быть свободным и захваченным, завершенным и активным.

*6. Какие способы переключения задач используются в ОС?*

* Круговая (round-robin).Управление между задачами производится в в каком-либо фиксированном порядке через определённые промежутки времени;
* Случайная (random).Через каждый определённый промежуток времени управление передаётся на произвольную задачу;
* Приоритетная (priority-driven).Каждая задача имеет уровень приоритета и управление получает незаблокированная задача имеющая наибольший уровень;
* Управляемая дедлайнами (deadline-driven). Передача управления производится в порядке, обеспечивающем выполнение задач в определённый срок.

*7.Объясните суть параметров, входящих в вызов pthread\_create().*

*int pthread\_create(pthread\_t \*thread,*

*const pthread\_attr\_t \*attr,*

*void \*(\*start\_routine) (void \*),*

*void \*arg),*

где:

*thread* – указатель на идентификатор потока;

*attr* – указатель на структуру данных, описывающих атрибуты потока;

*start\_routine* – имя функции, выполняющей роль потока;

*arg* - указатель на структуру данных, описывающих передаваемые в поток параметры.

*8. Объясните суть параметров, входящих в вызов pthread\_join().*

*int pthread\_join(pthread\_t thread, void \*\*retval),*

где:

*thread* – идентификатор потока;

*retval* – код завершения потока, переданный через функцию *pthread\_exit*.

*9. Опишите трассу выполнения программы.*

**объявление флагов завершения потоков;**

**объявление функций потоков;**

**основная программа**

**{**

**объявление идентификаторов потоков;**

**создание потоков из функций потоков;**

**ожидание нажатия клавиши;**

**установка флагов завершения потоков;**

**ожидание завершения потоков;**

**}**

**Выводы.**

В соответствии с поставленными задачами мною было разработано ПО, осуществляющее работу с потоками.

**Код программы**

#include<iostream>

#include<pthread.h>

#include<unistd.h>

using namespace std;

void\* echo\_1(void \*args)

{

cout << "first thread started \n" << flush;

int &flag\_adr = \*(int\*)args;

while (flag\_adr != 1)

{

cout << "1" << flush;

usleep(10000);

}

string \*retcode = new string("ok");

cout << "first thread ended with code : " << \*retcode << "\n" << flush;

pthread\_exit(retcode);

}

void\* echo\_2(void \*args)

{

cout << "second thread started \n" << flush;

int &flag\_adr = \*(int\*)args;

while (flag\_adr != 1)

{

cout << "2" << flush;

usleep(10000);

}

string \*retcode = new string("ok");

cout << "second thread ended with code : " << \*retcode << "\n" << flush;

pthread\_exit(retcode);

}

int main()

{

int first\_flag = 0,

second\_flag = 0;

pthread\_t thread\_1;

pthread\_t thread\_2;

int status\_1 = pthread\_create(&thread\_1, NULL, echo\_1, &first\_flag);

int status\_2 = pthread\_create(&thread\_2, NULL, echo\_2, &second\_flag);

int status\_adr;

int a;

cin >> a;

first\_flag = 1;

second\_flag = 1;

status\_1 = pthread\_join(thread\_1, (void\*\*)&status\_adr);

status\_2 = pthread\_join(thread\_2, (void\*\*)&status\_adr);

cout << "Main ended \n" << flush;

return 0;

}