**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра информационных систем**

отчет

**по лабораторной работе №2**

**по дисциплине «Операционные системы»**

Тема: «**СИНХРОНИЗАЦИЯ ПОТОКОВ С ПОМОЩЬЮ МЬЮТЕКСОВ И НЕИМЕНОВАННЫХ СЕМАФОРОВ**

**»**

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

Санкт-Петербург

2020

**Цель работы**

знакомство со средствами синхронизации потоков - двоичными и общими неименованными семафорами и с системными вызовами, обеспечивающими создание, закрытие, захват и освобождение мьютексов и неименованных семафоров.

**Основные теоретические положения**

Для обеспечения взаимного исключения при доступе нескольких потоков к одному критическому ресурсу используются семафоры.

Семафоры можно разделить на двоичные и общие, а также на неименованные и именованные.

Двоичный семафор – мьютекс – обладает только двумя состояниями – захвачен и свободен.

Если критический участок свободен, то поток выполняет операцию захвата мьютекса и входит в критический участок. При выходе из критического участка поток освобождает мьютекс.

Если критический участок занят, то поток, выполняя операцию захвата мьютекса, блокируется и не входит в критический участок. Активизация заблокированного потока и вход в критический участок происходит тогда, когда ранее вошедший в критический участок поток выходит из него и освобождает мьютекс.

Семафор отличается от мьютекса большим числом состояний за счет использования внутреннего счетчика. Это позволяет обеспечить большее разнообразие правил нахождения потоков в критическом участке.

При начальном состоянии счетчика семафора, равном 1, семафор эквивалентен мьютексу.

Мьютексы и неименованные семафоры используются для синхронизации потоков в рамках одного процесса.

**Экспериментальные результаты**

Вывод программы с использованием pthread\_mutex\_lock()

*ilya@ilya-VirtualBox:~/Документы/pr/pr\_labs/labs/lab2$ ./l1*

*first thread started*

*1first thread started*

*1111111112222222222*

*first thread ended with code : ok*

*second thread ended with code : ok*

*1*

*Main ended*

Вывод программы с использованием pthread\_mutex\_trylock()

*ilya@ilya-VirtualBox:~/Документы/pr/pr\_labs/labs/lab2$ ./l2*

*first thread started*

*1second thread started*

*second thread is blocked*

*111111111*

*second thread ended with code : ok*

*first thread ended with code : ok*

*0*

*Main ended*

Вывод программы с использованием pthread\_mutex\_timedlock()

*ilya@ilya-VirtualBox:~/Документы/pr/pr\_labs/labs/lab2$ ./l3*

*first thread started*

*1second thread started*

*111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111*

*2222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222*

*first thread ended with code : ok*

*second thread ended with code : ok*

*0*

*Main ended*

**Ответы на вопросы**

*Какой ресурс называется критическим?*

Под критическим разумеется некоторый программный либо аппаратный ресурс, который в каждый момент времени может использоваться одним и только одним потоком.

*Какой участок программы называется критическим?*

Участок программы между lock() и unlock()

*Какой режим программы называется режимом взаимного исключения?*

Режим программы, при котором один поток исполнения никогда не входит в критическую секцию одновременно с тем, как другой поток исполнения вошел в свою критическую секцию.

*Опишите алгоритмы захвата и освобождения мьютекса*

1. Объявить мьютекс, чтобы у необходимых потоков был к нему доступ
2. Инициализировать мьютекс
3. В необходимом потоке захватиь мьютекс
4. В этом же потоке осовбодить его
5. Выполнить для всех необходимых потоков
6. «Уничтожить мьютекс»

*Опишите алгоритм операций захвата и освобождения семафора*

1. Объявить семафор, чтобы у потоков был к нему доступ
2. Инициализировать семафор
3. Использовать sem\_wait() и sem\_post() по аналогии с lock() и unlock()
4. «Уничтожить» семафор

*Какими операциями с мьютексом и с неименованным семафором  
можно осуществить проверку занятости ресурса без блокирования  
потока?*

Для мьютексов int pthread\_mutex\_trylock(pthread\_mutex\_t \*mutex);

Для семафоров int sem\_getvalue(sem\_t \*sem, int \*sval);

**Выводы.**

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!11

**Код программы**

Код программы с использованием pthread\_mutex\_lock()

#include<iostream>

#include<pthread.h>

#include<unistd.h>

using namespace std;

pthread\_mutex\_t mutex;

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

{

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

pthread\_mutex\_lock(&mutex);

for(int i = 0; i < 10; i++)

{

cout << "1" << flush;

usleep(100);

}

pthread\_mutex\_unlock(&mutex);

sleep(1);

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

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

pthread\_exit(retcode);

}

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

{

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

pthread\_mutex\_lock(&mutex);

for(int i = 0; i < 10; i++)

{

cout << "2"<< flush;

usleep(100);

}

pthread\_mutex\_unlock(&mutex);

sleep(1);

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\_mutex\_init(&mutex, NULL);

pthread\_t thread\_1;

pthread\_t thread\_2;

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

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

int status\_adr;

int a;

cin >> a;

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

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

pthread\_mutex\_destroy(&mutex);

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

return 0;

}

Код программы с использованием pthread\_mutex\_trylock()

#include<iostream>

#include<pthread.h>

#include<unistd.h>

using namespace std;

pthread\_mutex\_t mutex;

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

{

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

if (pthread\_mutex\_trylock(&mutex) == 0)

{

for(int i = 0; i < 10; i++)

{

cout << "1";

usleep(100);

}

}

else

cout << "first thread is blocked\n";

cout << endl;

pthread\_mutex\_unlock(&mutex);

sleep(1);

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;

if (pthread\_mutex\_trylock(&mutex) == 0)

{

for(int i = 0; i < 10; i++)

{

cout << "2";

usleep(100);

}

}

else

cout << "second thread is blocked\n";

cout << endl;

pthread\_mutex\_unlock(&mutex);

sleep(1);

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\_mutex\_init(&mutex, NULL);

pthread\_t thread\_1;

pthread\_t thread\_2;

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

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

int status\_adr;

int a;

cin >> a;

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

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

pthread\_mutex\_destroy(&mutex);

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

return 0;

}

Код программы с использованием pthread\_mutex\_timedlock()

#include<iostream>

#include<pthread.h>

#include<unistd.h>

using namespace std;

pthread\_mutex\_t mutex;

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

{

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

timespec timeoutTime;

clock\_gettime(CLOCK\_REALTIME, &timeoutTime);

timeoutTime.tv\_sec += 1;

int retVal = pthread\_mutex\_timedlock(&mutex, &timeoutTime);

for(int i = 0; i < 100; i++)

{

cout << "1" << flush;

usleep(100);

}

cout << endl;

pthread\_mutex\_unlock(&mutex);

sleep(1);

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;

timespec timeoutTime;

clock\_gettime(CLOCK\_REALTIME, &timeoutTime);

timeoutTime.tv\_sec += 1;

int retVal = pthread\_mutex\_timedlock(&mutex, &timeoutTime);

for(int i = 0; i < 100; i++)

{

cout << "2" << flush;

usleep(100);

}

cout << endl;

pthread\_mutex\_unlock(&mutex);

sleep(1);

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\_mutex\_init(&mutex, NULL);

pthread\_t thread\_1;

pthread\_t thread\_2;

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

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

int status\_adr;

int a;

cin >> a;

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

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

pthread\_mutex\_destroy(&mutex);

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

return 0;

}