**ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ**

Государственное образовательное учреждение

высшего профессионального образования

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**МОСКОВСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**СВЯЗИ И ИНФОРМАТИКИ**

Кафедра Математической кибернетики и информационных технологий

Лабораторная работа №3

на тему «Знакомство с потоками и их синхронизацией в ОС Unix»

по дисциплине «Современные операционные системы»

Выполнила: студ. гр. ЗМПП1901

Иванова Е.В.

Проверила: И.о. зам. зав. кафедрой Мосева М.С.

Москва 2019

**Цель работы:** ознакомиться с подсистемой управления потоками в операционной системе Unix и основными программными средствами для создания, управления и удаления потоков.

**Задание:** изучить основные программные средства управления потоками ОС Unix, а также способы синхронизации потоков. Разработать приложения для многопоточных вычислений с использованием синхронизации посредством мьютексов, семафоров и условных переменных.

**Выполнение:**

**1. Синхронизированная работа потоков в критической секции с использованием мьютексов.**

#include <fstream>

#include <stdlib.h>

#include <stdio.h>

#include <pthread.h>

#include <string.h>

std::ofstream file;

pthread\_mutex\_t lock;

void\* func1(void\* arg);

void\* func2(void\* arg);

void\* func1(void\* arg){

pthread\_mutex\_lock(&lock);

for (int i=0;i<7;i++){

file<<"1 ";

}

pthread\_mutex\_unlock(&lock);

return NULL;

}

void\* func2(void\* arg){

pthread\_mutex\_lock(&lock);

for (int i=0;i<7;i++){

file<<"2 ";

}

pthread\_mutex\_unlock(&lock);

return NULL;

}

int main(){

file.open("file.txt");

pthread\_mutex\_init(&lock, NULL);

pthread\_t thread1, thread2;

pthread\_create(&thread1,NULL,&func1,NULL);

pthread\_join(thread1,NULL);

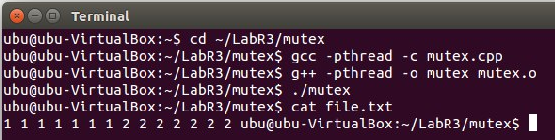
pthread\_create(&thread2,NULL,&func2,NULL);

pthread\_join(thread2,NULL);

pthread\_mutex\_destroy(&lock);

file.close();

}

****

**2. Синхронизированная работа потоков в критической секции с использованием семафоров.**

#include <semaphore.h>

#include <fstream>

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

#include <pthread.h>

int n;

sem\_t semafor;

void\* function(void\* a){

int b =(int)a;

sem\_wait(&semafor);

for (int i=0;i<5;i++){

n=n+b;

printf("%i ",n);

}

sem\_post(&semafor);

return NULL;

}

int main(){

sem\_init(&semafor,0,0);

sem\_post(&semafor);

pthread\_t thread1,thread2;

pthread\_create(&thread1,NULL,&function,(void\*)20);

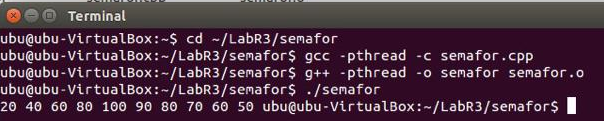
pthread\_join(thread1,NULL);

pthread\_create(&thread2,NULL,&function,(void\*)-10);

pthread\_join(thread2,NULL);

sem\_destroy(&semafor);

}

****

**3. Синхронизированная работа потоков в критической секции с использованием условных переменных.**

#include <fstream>

#include <stdlib.h>

#include <stdio.h>

#include <pthread.h>

#include <string.h>

void\* pluss(void \*a);

void\* minus(void \*a);

pthread\_mutex\_t mutex;

pthread\_cond\_t per;

int b = 6;

void\* pluss(void \*a){

for (int i = 0; i <= 5; i++){

pthread\_mutex\_lock(&mutex);

printf("%i ", b);

printf(" ");

b=b+4;

pthread\_cond\_signal(&per);

pthread\_cond\_wait(&per, &mutex);

pthread\_mutex\_unlock(&mutex);

}

}

void\* minus(void \*a){

while(true){

pthread\_mutex\_lock(&mutex);

printf("%i ",b);

printf(" ");

b=b-1;

pthread\_cond\_signal(&per);

pthread\_cond\_wait(&per, &mutex);

pthread\_mutex\_unlock(&mutex);

}

}

int main(){

pthread\_mutex\_init(&mutex,0);

pthread\_cond\_init(&per,0);

pthread\_t thread1;

pthread\_create(&thread1,NULL,pluss,NULL);

pthread\_t thread2;

pthread\_create(&thread2,NULL,minus,NULL);

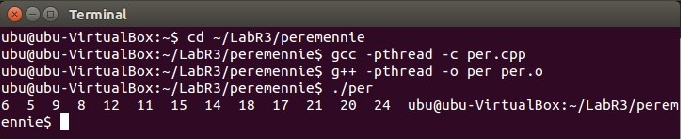
pthread\_join(thread1, NULL);

pthread\_cancel(thread2);

pthread\_mutex\_destroy(&mutex);

pthread\_cond\_destroy(&per);

}



**4. Без использования средств синхронизации.**

**4.1. Без условных переменных.**

#include <fstream>

#include <stdlib.h>

#include <stdio.h>

#include <pthread.h>

#include <string.h>

int b=6;

void\* pluss(void \*a);

void\* minus(void \*a);

int main(){

pthread\_t thread1;

pthread\_create(&thread1,NULL,pluss,NULL);

pthread\_t thread2;

pthread\_create(&thread2,NULL,minus,NULL);

pthread\_join(thread1, NULL);

pthread\_cancel(thread2);

}

void\* pluss(void \*a){

for (int i = 0; i <= 5; i++){

printf("%i ",b);

printf(" ");

b=b+4;

}

}

void\* minus(void \*a){

while (b>0){

printf("%i ",b);

printf(" ");

b=b-1;

}}

