**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра вычислительной техники**

отчет

**по лабораторной работе №4**

**по дисциплине «Программирование в среде Linux»**

Тема: «Управление потоками»

|  |  |  |
| --- | --- | --- |
| Студент гр. 6307 |  | Лазарев С. О. |
| Преподаватель |  | Разумовский Г. В. |

Санкт-Петербург

2019

# ЦЕЛЬ

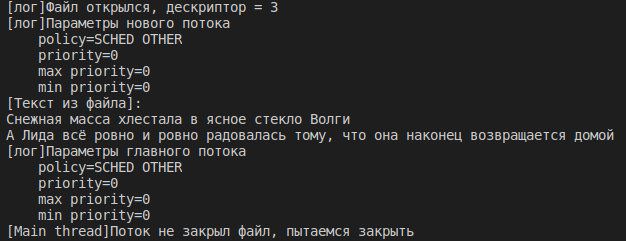
Знакомство с организацией потоков и способами синхронизации предков и потомков.

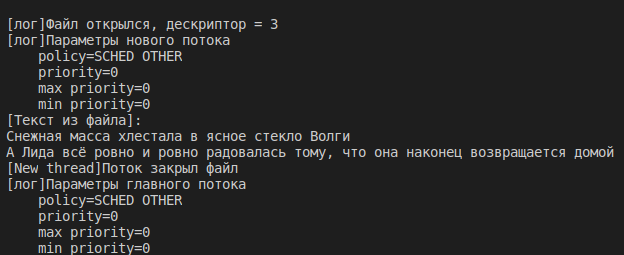
# ЗАДАНИЕ

Разработать программу, которая открывает текстовый файл, порождает поток, а затем ожидает его завершения. Потоку в качестве параметра передается дескриптор файла. Поток выводит на экран класс планирования, текущий, минимальный и максимальный приоритеты, содержимое файла и закрывает файл. После завершения работы потока программа должна вывести текущий приоритет и проверить — закрыт ли файл, и если он не закрыт, то принудительно закрыть. Результат проверки должен быть выведен на экран.

# РЕЗУЛЬТАТ

Поток не закрыл файл



Поток закрыл файл

# КОД

main.cpp

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include <fcntl.h>

#include <unistd.h>

#include <errno.h>

#include <iostream>

using namespace std;

int fd = -1;

static void display\_sched\_attr(int policy, struct sched\_param \*param) {

printf(" policy=%s\n priority=%d\n max priority=%d\n min priority=%d\n",

(policy == SCHED\_FIFO) ? "SCHED\_FIFO" :

(policy == SCHED\_RR) ? "SCHED\_RR" :

(policy == SCHED\_OTHER) ? "SCHED\_OTHER" :

"???",

param->sched\_priority,

sched\_get\_priority\_max(policy),

sched\_get\_priority\_min(policy)

);

}

static void display\_thread\_sched\_attr(const char\* msg) {

int policy;

struct sched\_param param;

printf("%s\n", msg);

pthread\_getschedparam(pthread\_self(), &policy, &param);

display\_sched\_attr(policy, &param);

}

int fd\_is\_valid(int fd)

{

return fcntl(fd, F\_GETFD) != -1 || errno != EBADF;

}

static void \* thread\_start(void \*arg) {

display\_thread\_sched\_attr("[лог]Параметры нового потока");

int fd = \*((int \*) arg);

char buf[1024];

size\_t nbytes;

ssize\_t bytes\_read;

nbytes = sizeof(buf);

bytes\_read = read(fd, buf, nbytes);

printf("[Текст из файла]:\n%s",buf);

//close(fd);

if(fd\_is\_valid(fd) == 0){

printf("[New thread]Поток закрыл файл\n");

}

pthread\_exit(0);

}

int main() {

int policy;

pthread\_t thread;

pthread\_attr\_t attr;

struct sched\_param param;

fd = open("/home/sergey/Рабочий стол/linux\_labs/lab4/test.txt", O\_RDONLY);

if(fd == -1){

printf("[лог]Файл не открылся\n");

return -1;

}else{

printf("[лог]Файл открылся, дескриптор = %d\n", fd);

}

pthread\_attr\_init(&attr);

pthread\_create(&thread, &attr, &thread\_start, &fd);

pthread\_join(thread, NULL);

display\_thread\_sched\_attr("[лог]Параметры главного потока");

if(fd\_is\_valid(fd) == 1){

printf("[Main thread]Поток не закрыл файл, пытаемся закрыть\n");

close(fd);

if(fd\_is\_valid(fd)){

printf("[Main thread]Всё равно не удалось закрыть файл\n");

}

else{

printf("[Main thread]Теперь файл закрыт");

}

}

return 0;

}

# ВЫВОДЫ

Получил опыт работы с потоками в ubuntu. Применил функции pthread\_create для создания потока, pthread\_join – для синхронизации предка и потомка.