Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное учреждение высшего образования

«Новгородский государственный университет имени Ярослава Мудрого»

Институт «Электронных и информационных систем»

Кафедра «Информационных систем и технологий»

**ОРГАНИЗАЦИЯ РАБОТЫ С РАЗДЕЛЯЕМОЙ ПАМЯТЬЮ. ПОНЯТИЕ ПОТОКОВ В UNIX**

Лабораторная работа №3 по учебной дисциплине «Операционные системы»

По направлению 09.03.01 «Информатика и вычислительная техника»

Отчёт

Принял преподаватель:

\_\_\_\_\_\_\_\_\_ Ананьев В. В.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2020 г.

Выполнил студент группы 8091:

\_\_\_\_\_\_\_ Бочкарев Б. А.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2020 г.

Великий Новгород

2020

**Цель работы:** научиться работать с разделяемой памятью и потоками в UNIX системах.

**Задание:**

## Задание по разделяемой памяти

Написать две программы на C или C++.

Первая программа должна сформировать массив из 20 случайных чисел. Для массива должна быть выделена область в разделяемой памяти. Затем эта программа должна запустить вторую программу и передать ей ключ для доступа к общей области памяти в качестве параметра командной строки.

Вторая программа должна прочитать из общего массива все числа и вывести их на экран.

Затем она должна отсортировать их по возрастанию и вывести результат сортировки на экран.

## Задание по потокам

Написать многопоточную программу. Основной поток (который начинается в функции main) должен породить два новых потока, дождаться их завершения и закончить работу. Первый из новых потоков должен вывести на экран "Hello Threads (1)", "Hello Threads (2)" и т.д., всего 10 раз. Второй поток должен выводить в цикле строки "This is iteration 1", "This is iteration 2", ... "This is iteration 12".

Каждый из порожденных потоков после вывода каждой строки должен делать системный вызов sleep() с параметром 1 для первого потока и параметром 2 для второго потока (чтобы можно было успевать наблюдать переключение между потоками).

В отчете привести исходные коды всех программ, а также результаты, выведенные ими на экран.

**Первая задача:**

Первая программа (receiver.cpp):

#include <iostream>

#include <sys/shm.h>

#include <stdlib.h>

using namespace std;

int comp(const void\*x, const void\* y)

{

return (\*((int\*)x) - \*((int\*)y));

}

int main(int argv, char \*argc[])

{

int memId = atoi(argc[1]);

const int AMOUNT = atoi(argc[2]);

int\* mem = (int\*)shmat(memId, 0, 0);

for (size\_t i = 0; i < AMOUNT; i++)

{

cout << mem[i] << " ";

}

cout << endl;

qsort(mem, AMOUNT, sizeof(int), comp);

for (size\_t i = 0; i < AMOUNT; i++)

{

cout << mem[i] << " ";

}

cout << endl;

return 0;

}

Вторая программа (sender.cpp):

#include <sys/shm.h>

#include <stdlib.h>

#include <stdio.h>

#include <ctime>

using namespace std;

const int AMOUNT = 20;

int main()

{

srand(time(NULL));

int memId = shmget(IPC\_PRIVATE, sizeof(int) \* AMOUNT, 0600|IPC\_CREAT|IPC\_EXCL);

int\* arr = (int\*)shmat(memId, 0, 0);

for (size\_t i = 0; i < AMOUNT; i++)

{

arr[i] = rand() % 51;

}

char callbuf[1024];

sprintf(callbuf, "./receiver %i %i", memId, AMOUNT);

system(callbuf);

return 0;

}

**Вторая задача:**

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include <unistd.h>

void \*task1()

{

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

{

printf("Hello Threads (%i), ", i + 1);

sleep(1);

}

}

void \*task2()

{

for (int i = 0; i < 12; i++)

{

printf("This is iteration %i, \n", i + 1);

sleep(2);

}

}

int main()

{

pthread\_t thread1, thread2;

int res1 = pthread\_create(&thread1, NULL, task1, NULL);

int res2 = pthread\_create(&thread2, NULL, task2, NULL);

pthread\_join(thread1, NULL);

pthread\_join(thread2, NULL);

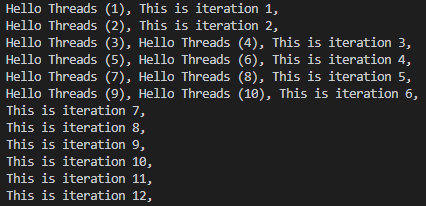
printf("\n");

return 0;

}

**Результаты:**





**Вывод:** В процессе выполнения лабораторной работы я научился работать с разделяемой памятью и потоками.