Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования "Новгородский государственный университет имени Ярослава Мудрого"

Кафедра «Информационных технологий и систем»

Дисциплина «Операционные системы»

Отчет по лабораторной работе

**«Средства System V IPC. Организация работы с разделяемой памятью. Понятие потоков в UNIX»**

Выполнил студент группы 9091

\_\_\_\_\_\_\_\_\_\_\_\_\_/Боговаров Арсений Алексеевич

Принял преподаватель

\_\_\_\_\_\_\_\_\_\_\_\_/Ананьев Владислав Валерьевич

Великий Новгород

2021

# 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".

# **Ход работы:**

1) Создаю пустую папку :

mkdir 3\_laba

cd 3\_laba

2) Запустить редактор vim:

Разделяемая память:

1. Создаю файл sender.c :

#include <stdio.h>

#include <sys/shm.h>

#include <stdlib.h>

#include <time.h>

int main()

{

const size\_t memSize = sizeof(int) \* 20;

int memId = shmget(IPC\_PRIVATE, memSize, 0600 | IPC\_CREAT | IPC\_EXCL);

printf("shmid = %i\n", memId);

if (memId <= 0)

{

printf("error with shmget()\n");

return -1;

}

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

if (NULL == mem)

{

printf("error with shmat()\n");

return -2;

}

srand(time(NULL));

for (int i = 0; i < 20; i++)

mem[i] = rand() % 100;

char callbuf[1024];

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

system(callbuf);

return 0;

}

1. Создаю файл receiver.c:

#include <stdio.h>

#include <sys/shm.h>

#include <stdlib.h>

int comp(const int \*a, const int \*b)

{

return \*a - \*b;

}

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

{

if (argv <= 1)

{

printf("not enough params\n");

return -1;

}

char \*paramStr = argc[1];

int memId = atoi(paramStr);

if (memId == 0)

{

printf("incorrect parameter string: %s\n", paramStr);

return -2;

}

printf("receiving the memory data: shmid = %i\n", memId);

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

if (NULL == mem)

{

printf("error with shmat()\n");

return -3;

}

for (int i = 0; i < 20; i++)

printf("%i ", mem[i]);

printf("\n");

qsort(mem, 20, sizeof(int), comp);

for (int i = 0; i < 20; i++)

printf("%i ", mem[i]);

printf("\n");

return 0;

}

1. Создать файл threader.c со следующим содержимым:

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

struct data

{

const char \*text;

int count;

int time;

};

void \*print\_message\_function(void \*data\_in)

{

struct data \*input = (struct data \*)data\_in;

for (int i = 0; i < input->count; i++)

{

printf(input->text, i);

sleep(input->time);

}

return (void \*)(0);

}

int main()

{

pthread\_t thread1, thread2;

struct data data1 = {"Hello Threads (%i)\n", 10, 1};

struct data data2 = {"This is iteration %i\n", 12, 2};

int res1 = pthread\_create(&thread1, NULL, print\_message\_function, (void \*)&data1);

int res2 = pthread\_create(&thread2, NULL, print\_message\_function, (void \*)&data2);

int iret1, iret2;

pthread\_join(thread1, (void \*\*)&iret1);

pthread\_join(thread2, (void \*\*)&iret2);

printf("Thread 1 returns: %d\n", iret1);

printf("Thread 2 returns: %d\n", iret2);

return 0;

}

1. Компилирую созданные файлы и запускаю программы:

all:

echo "Задание по разделяемой памяти"

gcc sender.c -o sender

gcc receiver.c -o receiver

./sender

echo "Задание по потокам"

gcc threader.c -lpthread -o threader

./threader

# **Вывод**

Таким образом я научился работать с потоками, нитями и разделяемой памятью.