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»

**Средства System V IPC.**

**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.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2021 г.

Выполнил студент группы 9091:

\_\_\_\_\_\_\_ Сухарев И.Д.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2021 г.

Великий Новгород

2020

**Цель работы:** познакомиться с механизмами разделённой памяти и многопоточности.

**Задания:**

1. 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.

1. 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).

**[sender.c]:**

#include "stdio.h"

#include "sys/shm.h"

#include "stdlib.h"

#include "sys/ipc.h"

#include "time.h"

int main(void)

{

const size\_t memSize = 80;

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

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

if (memId <= 0)

{

printf("error with shmid()\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;

}

**[receiver.c]:**

#include "stdio.h"

#include "sys/shm.h"

#include "stdlib.h"

void print\_array\_int(int\* arr)

{

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

printf("%i ", \*(arr + i));

printf("\n");

}

int compare\_int\_value(const void\* a, const void\* b)

{

return \*((int\*)a) - \*((int\*)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;

}

printf("receiving next array:\n");

print\_array\_int(mem);

qsort(mem, 20, 4, compare\_int\_value);

printf("sorted array:\n");

print\_array\_int(mem);

return 0;

}

**Результат, выведенный на экран:**

/ilya/Desktop/laba3/shared\_memory$ make  
gcc sender.c -o sender  
gcc receiver.c -o receiver  
./sender  
6523 6173 8092 3401 250 9125 7923 6249 7146 557 2065 6948 277 6650 666 260 5590 984 6163 5738  
250 260 277 557 666 984 2065 3401 5590 5738 6163 6173 6249 6523 6650 6948 7146 7923 8092 9125

**[threader.c]:**

#include "stdio.h"

#include "stdlib.h"

#include "pthread.h"

void\* print\_message\_function1()

{

for (int i = 1; i < 11; i++)

printf("Hello Thread(%i)\n", i), sleep(1);

return NULL;

}

void\* print\_message\_function2()

{

for (int i = 1; i < 13; i++)

printf("This is iteration %i\n", i), sleep(2);

return NULL;

}

int main(void)

{

pthread\_t thread1, thread2;

int res1 = pthread\_create(&thread1, NULL, print\_message\_function1, NULL);

int res2 = pthread\_create(&thread2, NULL, print\_message\_function2, NULL);

int iret1, iret2;

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

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

return 0;

}

**Результат, выведенный на экран:**

/ilya/Desktop/laba3/threads$ make  
gcc threader.c -lpthread -o threader  
./threader  
Hello Threads (1), This is iteration 1, Hello Threads (2), Hello Threads (3), This is iteration 2, Hello Threa  
ds (4), This is iteration 3, Hello Threads (5), Hello Threads (6), This is iteration 4, Hello Threads (7), Hel  
lo Threads (8), This is iteration 5, Hello Threads (9), Hello Threads (10), This is iteration 6, This is itera  
tion 7, This is iteration 8, This is iteration 9, This is iteration 10, This is iteration 11, This is iteratio  
n 12

**Вывод:** В процессе взаимодействия с теорией лабораторной работы №3 я узнал о механизации разделенной памяти, а также о многопоточности в системах UNIX.