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:

\_\_\_\_\_\_\_ Кудрявцев В. В.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2021 г.

Великий Новгород

2021

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

**Задание:**

1. Написать две программы на C или C++.
   1. 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.
   2. 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.
2. 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).

**Содержание файла 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;

}

**Содержание файла 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;

}

**Содержание файла threader.c:**

#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;

}

**Результат, выведенный на экран:**

make

echo "Задание по разделяемой памяти"

Задание по разделяемой памяти

gcc sender.c -o sender

gcc receiver.c -o receiver

receiver.c: In function ‘main’:

receiver.c:40:30: warning: passing argument 4 of ‘qsort’ from incompatible pointer type [-Wincompatible-pointer-types]

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

^~~~

In file included from receiver.c:3:0:

/usr/include/stdlib.h:827:13: note: expected ‘\_\_compar\_fn\_t {aka int (\*)(const void \*, const void \*)}’ but argument is of type ‘int (\*)(const int \*, const int \*)’

extern void qsort (void \*\_\_base, size\_t \_\_nmemb, size\_t \_\_size,

^~~~~

./sender

shmid = 15

receiving the memory data: shmid = 15

28 34 41 10 25 21 95 18 57 14 17 35 46 79 2 57 77 76 81 65

2 10 14 17 18 21 25 28 34 35 41 46 57 57 65 76 77 79 81 95

echo "Задание по потокам"

Задание по потокам

gcc threader.c -lpthread -o threader

threader.c: In function ‘print\_message\_function’:

threader.c:18:3: warning: implicit declaration of function ‘sleep’ [-Wimplicit-function-declaration]

sleep(input->time);

^~~~~

./threader

Hello Threads (0)

This is iteration 0

Hello Threads (1)

This is iteration 1

Hello Threads (2)

Hello Threads (3)

This is iteration 2

Hello Threads (4)

Hello Threads (5)

This is iteration 3

Hello Threads (6)

Hello Threads (7)

This is iteration 4

Hello Threads (8)

Hello Threads (9)

This is iteration 5

This is iteration 6

This is iteration 7

This is iteration 8

This is iteration 9

This is iteration 10

This is iteration 11

Thread 1 returns: 0

Thread 2 returns: 0

**Вывод:** В процессе выполнения лабораторной работы я на практике познакомился с механизмами разделённой памяти и многопоточности в UNIX.