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.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2020 г.

Выполнил студент группы 8091:

\_\_\_\_\_\_\_ Кузин И. А.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2020 г.

Великий Новгород

2020

**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()

{

srand((unsigned)(time(0)));

int memId = shmget(IPC\_PRIVATE, sizeof(int) \* 20, 0600|IPC\_CREAT|IPC\_EXCL);

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

for (size\_t i = 0;i < 20;i++)

{

numbers[i] = rand() % 10000;

}

char callbuf[1024];

sprintf(callbuf, "./receiver %i %i", memId, 20);

system(callbuf);

return 0;

}

**Содержание файла receiver.c:**

#include <stdio.h>

#include <sys/shm.h>

#include <stdlib.h>

int compare\_ints(const void\* a, const void\* b)

{

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

}

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

{

int memId = atoi(argc[1]);

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

for (size\_t i = 0;i < atoi(argc[2]);i++)

{

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

}

printf("\n");

qsort(mem, atoi(argc[2]), sizeof(int), compare\_ints);

for (size\_t i = 0;i < atoi(argc[2]);i++)

{

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

}

printf("\n");

return 0;

}

**Результат, выведенный на экран:**

|  |
| --- |
| kuzin@DESKTOP-2I40N54:~/os-labs/3/shared\_memory$ make |
|  |

|  |
| --- |
| gcc sender.c -o sender |
|  |

|  |
| --- |
| gcc receiver.c -o receiver |
|  |

|  |
| --- |
| ./sender |
|  |

|  |
| --- |
| 4803 8515 1248 4443 8568 5110 3650 7432 927 4955 9683 157 8991 434 1549 2408 2659 7753 7746 5349 |
|  |

157 434 927 1248 1549 2408 2659 3650 4443 4803 4955 5110 5349 7432 7746 7753 8515 8568 8991 9683

**Содержание файла threader.c:**

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include <unistd.h>

void \*task1()

{

for (int i = 1;i <= 10;i++)

{

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

sleep(1);

}

}

void \*task2()

{

for (int i = 1;i <= 12;i++)

{

printf("This is iteration %i, ", i);

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;

}

**Результат, выведенный на экран:**

kuzin@DESKTOP-2I40N54:~/os-labs/3/threads$ make

gcc threader.c -lpthread -o threader

./threader

Hello Threads (1), This is iteration 1, Hello Threads (2), This is iteration 2, Hello Threads (3), Hello Threads (4), This is iteration 3, Hello Threads (5), Hello Threads (6), This is iteration 4, Hello Threads (7), Hello Threads (8), This is iteration 5, Hello Threads (9), Hello Threads (10), This is iteration 6, This is iteration 7, This is iteration 8, This is iteration 9, This is iteration 10, This is iteration 11, This is iteration 12,

**Вывод:** В процессе выполнения данной лабораторной работы, я научился работать с механизмами разделённой памяти и многопоточности в UNIX.