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»

**Процессы в ОС UNIX**

Лабораторная работа №4 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:**

Объединить программы первой части предыдущей лабораторной в одну программу.

Вначале программа должна сгенерировать 20 случайных чисел, разместить их в разделяемой памяти и вывести на экран.

Затем требуется породить новый процесс, который выполнит сортировку самостоятельно, без запуска других программ (в отличие от предыдущей лабораторной).

Второй процесс должен отсортировать числа и также вывести их на экран.

Первый процесс должен дождаться завершения работы второго, после чего самостоятельно выполнить освобождение выделенной разделяемой памяти и всех прочих выделенных ресурсов (при наличии таковых).

**Содержание файла processes.c:**

#include <stdio.h>

#include <sys/wait.h>

#include <sys/shm.h>

#include <stdlib.h>

#include <unistd.h>

#include <time.h>

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

{

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

}

void sortAndPrint(int \*mem)

{

qsort(mem, 20, sizeof(int), compare\_ints);

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

{

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

}

printf("\n");

}

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;

}

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

{

printf("%d ", numbers[i]);

}

printf("\n");

int child\_id = fork();

if (child\_id == 0)

{

sortAndPrint(numbers);

}

else

{

waitpid(child\_id, NULL, 0);

}

shmdt(numbers);

return 0;

}

**Результат, выведенный на экран:**

kuzin@DESKTOP-2I40N54:~/os-labs/4$ make

gcc processes.c -o processes

./processes

9602 5612 6099 5134 8275 7000 3715 6859 4285 9288 7135 5809 2951 9911 9286 2064 2168 9903 2416 3173

2064 2168 2416 2951 3173 3715 4285 5134 5612 5809 6099 6859 7000 7135 8275 9286 9288 9602 9903 9911

**Вывод:** В процессе выполнения данной лабораторной работы, я на практике познакомился и научился работать с механизмами запуска новых процессов в UNIX.