Министерство образования и науки РФ

Федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

Новгородский государственный университет имени Ярослава Мудрого

Институт Электронных и Информационных систем

Кафедра «Информационных технологий и систем»

ОПЕРАЦИОННЫЕ СИСТЕМЫ

Лабораторная работа №4

Выполнил:

Студент Гр. №8091

Кулаков Игорь Юрьевич

Принял:

Ананьев Владислав Валерьевич

Великий Новгород

2020 г.

# Лабораторная работа №4 –«Процессы в ОС UNIX»

## Задание на лабораторную работу

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).

В отчете привести исходный код программы, а также результаты, выведенные на экран.

***Код:***

#include <stdio.h>

#include <sys/shm.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/wait.h>

void \*allocateSharedMemory(size\_t memSize, int &memId)

{

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

if (memId <= 0)

{

perror("error with memId");

return NULL;

}

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

if (NULL == mem)

{

perror("error with shmat");

}

return mem;

}

void childMainCode(int \*sharedMem) //Сортируем и выводим на экран

{

int key =0;

int i=0;

for(int j=1;j<20;j++){

key = sharedMem[j];

i=j-1;

while(i>=0 && sharedMem[i]>key){

sharedMem[i+1]=sharedMem[i];

i=i-1;

sharedMem[i+1]=key;

}

}

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

{

printf("%d\n", sharedMem[i]);

}

}

int main()

{

int memId;

int \*sharedMem = (int \*)allocateSharedMemory(1024, memId);

printf("memId = %d\n", memId);

printf("starting child process...\n");

pid\_t childId = fork();

//делаем копию процесса

if (childId < 0)

{

perror("error with fork()\n");

}

else if (childId > 0)

{

printf("Parent: This is main process. Child id = %d\n", childId);

printf("Parent: awaiting for child.\n");

int a[20];

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

sharedMem[i]=rand(); //Формируем массив случайных чисел

waitpid(childId, NULL, 0); //Ждем завершения дочернего процесса

shmdt(sharedMem); //удаляем разделенную память

printf("Shared memory is clear.\n");

}

else

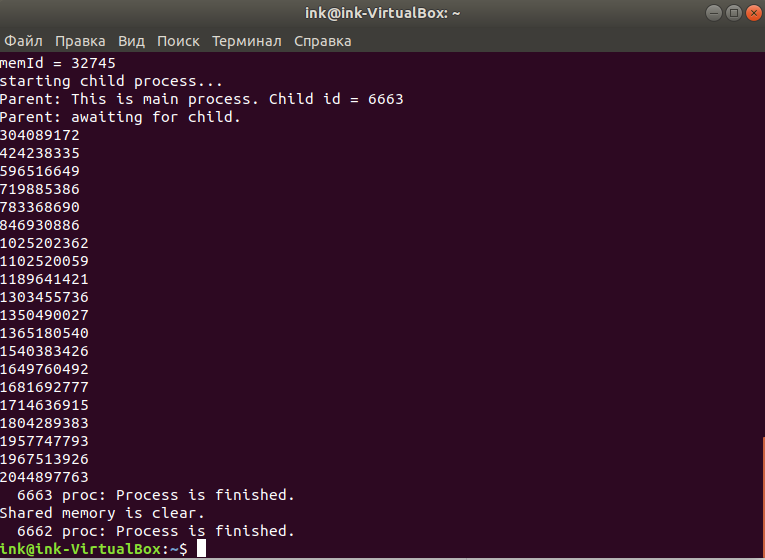
{

childMainCode(sharedMem);

}

pid\_t curId = getpid();

printf("%6d proc: Process is finished.\n", curId);



return 0;

}

***Работа программы:***