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»

**Организация взаимодействия процессов через pipe и FIFO в UNIX**

Лабораторная работа №2 по учебной дисциплине «Операционные системы»

По направлению 09.03.01 «Информатика и вычислительная техника»

Отчёт

Принял:

\_\_\_\_\_\_\_\_\_ Ананьев В. В.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2020 г.

Выполнил:

Студент группы 8091

\_\_\_\_\_\_\_ Баранов А. С.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2020 г.

Великий Новгород

2020

**Цель работы:** научиться пользоваться pipe и FIFO.

**Задание:**

Основной процесс должен сгенерировать N случайных чисел и вывести их на экран. Число N должно быть получено из параметров командной строки.

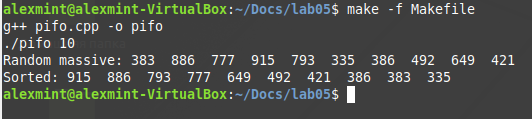
Затем требуется породить новый процесс, который выполнит сортировку.

Передать числа в него нужно через FIFO, а вернуть от дочернего в родительский через pipe.

Второй процесс должен отсортировать числа по убыванию.

Первый процесс в это время должен ожидать сообщения от второго процесса о готовности к передаче в обратном направлении результата работы. Сообщение можно передать через уже установленный pipe или FIFO, можно создать новый pipe или FIFO, или воспользоваться любым другим способом.

**Выполненные команды и вывод содержимого файлов в консоль:**



**Код программы:**

#include <iostream>

#include <stdio.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

using namespace std;

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

{

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

}

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

{

int n, result; size\_t size;

int fdp[2];

const char name[11] = "myfif.fifo";

(void)umask(0);

n = atoi(argc[1]);

int\* mas = new int[n];

cout << "Random massive: ";

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

{

mas[i] = rand()%1000;

cout << mas[i] << " ";

}

cout << endl;

if(mknod(name, S\_IFIFO | 0666, 0))

{

cout << "Can`t create FIFO" << endl;

exit(-1);

}

if (pipe(fdp) < 0)

{

cout << "Can`t creat the pipe!";

exit(-1);

}

size = sizeof(int)\*n;

result = fork();

if ( result < 0)

{

cout << "Can`t fork child." << endl;

exit(-1);

} if(result > 0)

{

close(fdp[1]);

int fifo = open(name, O\_WRONLY);

if ( fifo < 0)

{

cout << "Can`t open fifo." << endl;

}

int w = write(fifo, mas , size);

if(w != size)

{

cout << "Can`t write all numbers." << endl;

}

close(fifo);

int\* sorted\_mas = new int[n];

if (read(fdp[0],sorted\_mas, size) != size)

{

cout << "Can`t read all message." << endl;

}

close(fdp[0]);

cout << "Sorted: ";

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

{

cout << sorted\_mas[i] << " ";

}

cout << endl;

delete[] sorted\_mas;

delete[] mas;

remove("myfif.fifo");

}else

{

close(fdp[0]);

int r;

int fifo = open(name, O\_RDONLY);

if ( fifo < 0)

{

cout << "Can`t open fifo." << endl;

}

int\* rec\_mas = new int[n];

r = read(fifo,rec\_mas, size);

if ( r != size)

{

cout << "Can`t read all message." << endl;

}

close(fifo);

qsort(rec\_mas, n,sizeof(int), compare\_ints);

if (write(fdp[1],rec\_mas, size) != size)

{

cout << "Can`t write all numbers." << endl;

}

close(fdp[1]);

delete[] rec\_mas;

delete[] mas;

}

return 0;

}

**Вывод:** В ходе выполнения лабораторной работы я познакомился с основой работы c pipe и FIFO, передачей информации родственным процессам через fork.