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 №5 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:

\_\_\_\_\_\_\_ Григорьев Д. И.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2020 г.

Великий Новгород

2020

**Цель работы:** научиться работать с PIPE и FIFO в UNIX-like системах

**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 неименованный pipe (нечетным номерам - через FIFO).

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.

Для четных номеров студентов: Передача отсортированного массива в первый процесс должна быть организована через FIFO.

Для нечетных номеров студентов: Передача отсортированного массива в первый процесс должна быть организована через pipe.

Второй процесс не должен осуществлять никакого вывода на экран. После его завершения первый процесс должен вывести на экран отсортированный массив.

Первый процесс должен самостоятельно выполнить освобождение всех выделенных ресурсов перед завершением своей работы.

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

**Реализация:**

#include <stdio.h>

#include <sys/stat.h>

#include <time.h>

#include <stdlib.h>

#include <unistd.h>

#include <fcntl.h>

const char\* NAME = "testfile";

int comp(const void\* x, const void\* y)

{

return (\*((int\*)y) - \*((int\*)x));

}

int\* random\_dig(int n)

{

srand(time(NULL));

int \*arr = malloc(sizeof(int) \* n);

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

{

arr[i] = rand() % 51;

}

return arr;

}

void out\_dig(int\* arr, int n)

{

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

{

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

}

printf("\n");

}

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

{

int n = atoi(argc[1]);

int \*arr = random\_dig(n);

out\_dig(arr, n);

mknod(NAME, S\_IFIFO | 0666, 0);

int fd[2];

pipe(fd);

int child\_id = fork();

if (child\_id == 0)

{

close(fd[0]);

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

int \*sndarr = malloc(sizeof(int) \* n);

read(fifo, sndarr, sizeof(int) \* n);

close(fifo);

qsort(sndarr, n, sizeof(int), comp);

write(fd[1], sndarr, sizeof(int) \* n);

close(fd[1]);

}

else

{

close(fd[1]);

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

write(fifo, arr, sizeof(int) \* n);

close(fifo);

int \*sortarr = malloc(sizeof(int) \* n);

read(fd[0], sortarr, sizeof(int) \* n);

out\_dig(sortarr, n);

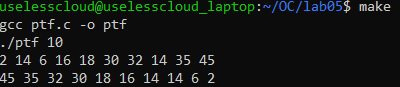
close(fd[0]);

unlink(NAME);

}

}

**Результат выполнения программы:**

****

**Вывод:** В процессе выполнения лабораторной работы я научился работать с PIPE и FIFO.