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:

\_\_\_\_\_\_\_ Кудряшов И. С.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2020 г.

Великий Новгород

2020

**Цель работы:** познакомиться с механизмами работы 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.

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.

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.

**Содержание файла pipe\_fifo.c:**

#include <stdio.h>

#include <sys/stat.h>

#include <stdlib.h>

#include <unistd.h>

#include <time.h>

#include <fcntl.h>

const char \*FIFO\_NAME = "test1";

int comp(const int \*a, const int \*b)

{

return \*a - \*b;

}

int \*get\_random\_nums(int n)

{

srand(time(NULL));

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

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

{

nums[i] = rand() % 100000;

}

return nums;

}

void print\_nums(int \*nums, int c)

{

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

{

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

}

printf("\n");

}

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

{

int n = atoi(argc[1]);

int \*nums = get\_random\_nums(n);

print\_nums(nums, n);

mknod(FIFO\_NAME, S\_IFIFO | 0666, 0);

int p[2];

pipe(p);

int child\_id = fork();

if (child\_id == 0)

{

close(p[0]);

int fifo = open(FIFO\_NAME, O\_RDONLY);

int \*received\_nums = malloc(sizeof(int) \* n);

read(fifo, received\_nums, sizeof(int) \* n);

close(fifo);

qsort(received\_nums, n, sizeof(int), comp);

write(p[1], received\_nums, sizeof(int) \* n);

close(p[1]);

free(received\_nums);

}

else

{

close(p[1]);

int fifo = open(FIFO\_NAME, O\_WRONLY);

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

close(fifo);

int \*sorted\_nums = malloc(sizeof(int) \* n);

read(p[0], sorted\_nums, sizeof(int) \* n);

print\_nums(sorted\_nums, n);

close(p[0]);

unlink(FIFO\_NAME);

free(sorted\_nums);

}

free(nums);

}

**Результат, выведенный на экран:**

make

gcc pipe\_fifo.c -o pipe\_fifo

pipe\_fifo.c: In function ‘main’:

pipe\_fifo.c:54:46: warning: passing argument 4 of ‘qsort’ from incompatible pointer type [-Wincompatible-pointer-types]

qsort(received\_nums, n, sizeof(int), comp);

^~~~

In file included from pipe\_fifo.c:3:0:

/usr/include/stdlib.h:827:13: note: expected ‘\_\_compar\_fn\_t {aka int (\*)(const void \*, const void \*)}’ but argument is of type ‘int (\*)(const int \*, const int \*)’

extern void qsort (void \*\_\_base, size\_t \_\_nmemb, size\_t \_\_size,

^~~~~

./pipe\_fifo 20

82902 72020 52759 22719 52660 19207 72660 37587 7924 51685 67960 85031 32895 85244 16891 78986 38641 45035 56998 62008

7924 16891 19207 22719 32895 37587 38641 45035 51685 52660 52759 56998 62008 67960 72020 72660 78986 82902 85031 85244

**Вывод:** В процессе выполнения лабораторной работы я на практике познакомился с работой pipe и FIFO в UNIX.