Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования «Новгородский Государственный Университет имени Ярослава Мудрого»

Кафедра информационных технологий и систем

Процессы в ОС UNIX

Лабораторная работа №5 по дисциплине

Операционные системы

Выполнили студенты гр. 8091

Кулаков Игорь Юрьевич

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/

Принял

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/

### Великий Новгород

### 2020

# Лабораторная работа №5 –«Организация взаимодействия процессов через pipe и FIFO в UNIX»

Перед началом выполнения задания требуется узнать свой порядковый номер в списке группы на novsu.ru. Если номер четный, то передача данных из первого процесса во второй должна быть осуществлена через pipe, а обратно через FIFO. Если номер нечетный, то наоборот - из первого процесса во второй через FIFO, а результат обратно через pipe.

## Задание на лабораторную работу

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.

(Замечание: все каналы передачи, которые потребуются в ходе работы, как pipe, так и FIFO, рекомендуется создать до порождения второго процесса.)

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.

***Код:***

#include <stdio.h>

#include <sys/shm.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <string.h>

#include <fcntl.h>

#include <algorithm>

#include <sys/stat.h>

bool comp(int a, int b)

{

return a>b?1:0;

}

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

{

int N;

N= atoi(argv[1]);

int pipefd[2];

pid\_t cpid;

int buf;

srand(120);

int fd;

size\_t size;

char name[]="my.fifo";

(void)umask(0);

if(mknod(name, S\_IFIFO|0666,0)<0) //создаем FIFO

{

printf("Cant FIFO");

exit(-1);

}

if (argc != 2) {

fprintf(stderr, "Usage: %s <string>\n", argv[0]);

exit(EXIT\_FAILURE);

}

if (pipe(pipefd) == -1) { //создаем pipe

perror("pipe");

exit(EXIT\_FAILURE);

}

cpid = fork(); //попрождаем второй процесс

if (cpid == -1) {

perror("fork");

exit(EXIT\_FAILURE);

}

if (cpid == 0) {

close(pipefd[1]);

int mes[1];

int mas[N];

int k=0;

while (read(pipefd[0], mes, 4) > 0) //получаем числа через pipe

{

mas[k]=mes[0];

// printf("%d",mas[k]);

k++;

}

std::sort(mas, mas+N, comp); //сортировка

//for (int i =0; i<N; i++)

// printf("%d",mas[i]);

close(pipefd[0]);

if((fd=open(name, O\_WRONLY))<0)

{

printf("Cant FIFO");

exit(-1);

}

//for (int i=0; i<N; i++)

//{

//mes[0]=mas[k];

write(fd, mas, 4\*N); //передаем массив с отсортированными числами через FIFO

//}

close(fd);

exit(0);

} else {

close(pipefd[0]);

for (int i=0; i<N; i++) //генирируем цисла и передаем их через pipe

{

int s[1];

s[0] =rand()%10;

write(pipefd[1], s, 4);

}

close(pipefd[1]);

if((fd=open(name, O\_RDONLY))<0)

{

printf("Cant FIFO");

exit(-1);

}

wait(NULL);

int s[1];

while (read(fd, s, 4) > 0) //получаем отсортированные числа и выводим их на экран

{

printf ("%d\n", s[0]);

}

close(fd);

unlink(name);

exit(0);

}

}