МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИМЕНИ И. С. ТУРГЕНЕВА»

Кафедра информационных систем

**ОТЧЕТ**

по лабораторной работе № 5

на тему: «Управление процессами»

по дисциплине: «Операционные системы»

Выполнили: Выполнил: Кубарев А.Н.

Институт приборостроения, автоматизации и информационных технологий

Направление: 09.03.03 «Прикладная информатика»

Группа: 71-ПИ

Проверил: Рыженков Д.В., Загородних Н.А.

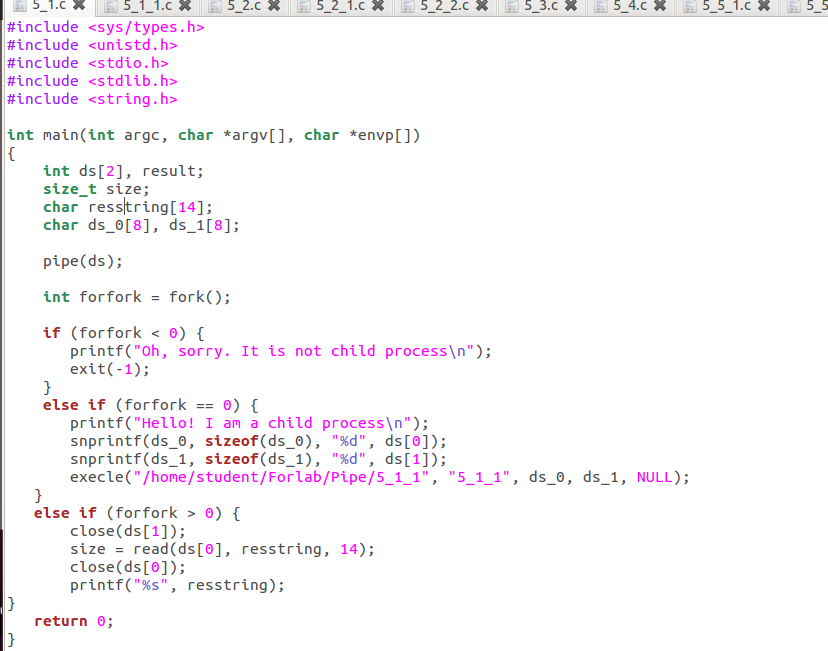
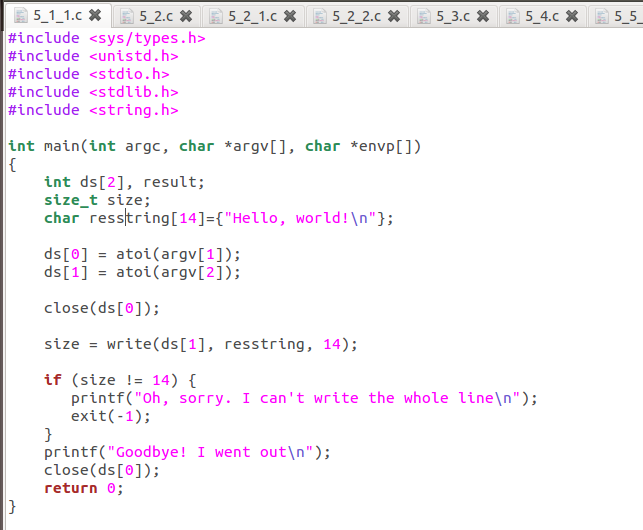
Отметка о зачете:

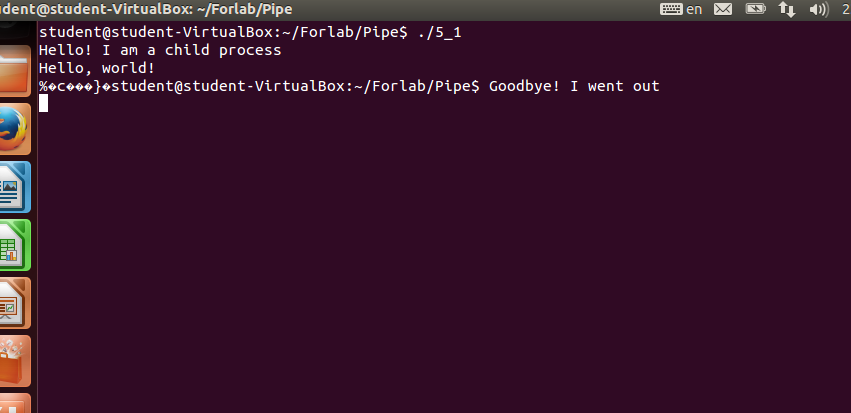
Дата: «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2017 г.

Орел, 2017 г.

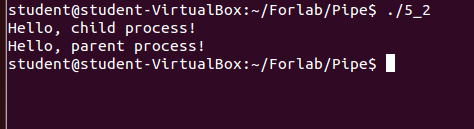
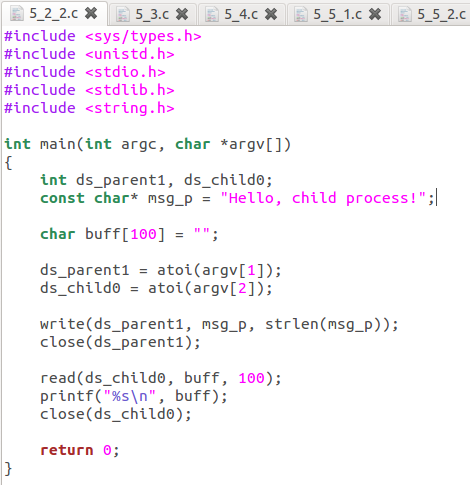
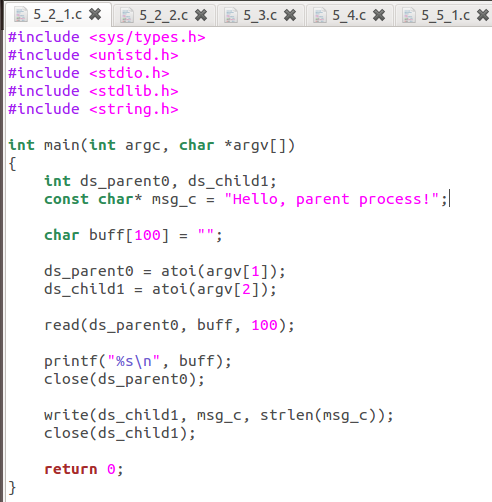
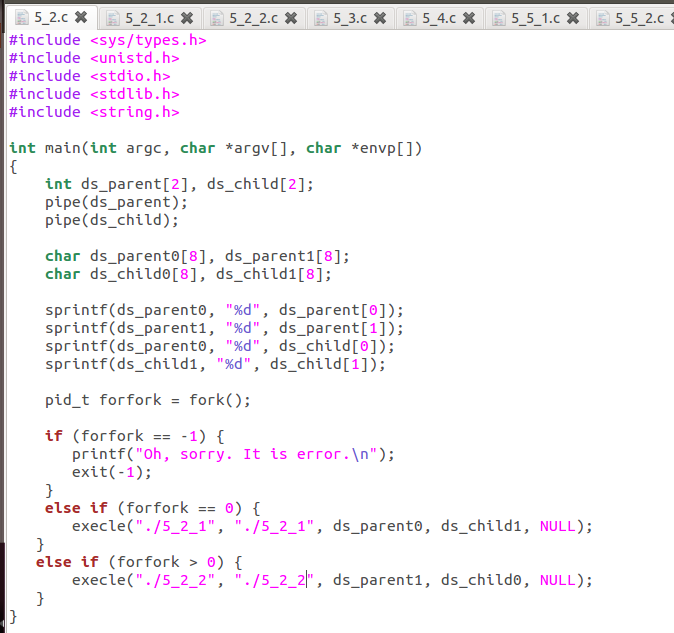
**Порядок выполнения работы**

**Задание 9:** модифицируйте этот пример для связи между собой двух родственных процессов, исполняющих разные программы.

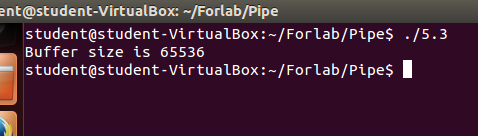
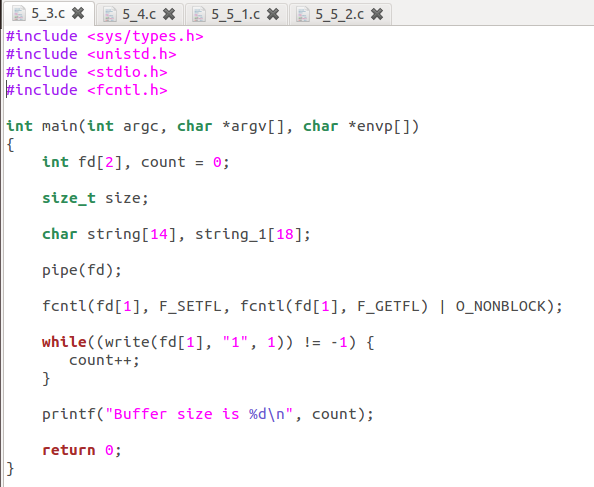




**Задание 10:** измените предыдущий пример, причем для двухсторонней связи используйте два канала: один для передачи от родителя ребенку, второй - от ребенка родителю.



**Задание 11:** напишите программу для определения в используемой операционной системе размера канала. Не забудьте, что при попытке записи в канал, в котором нет места, системный вызов write завершится с ошибкой.



**Задание 12:** Для иллюстрации взаимодействия родственных процессов через FIFO рассмотрите приведенную ниже программу.

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <unistd.h>

#include <stdio.h>

int main(){

int fd, result;

size\_t size;

char resstring[14];

char name[]="aaa.fifo";

/\* Обнуляем маску создания файлов текущего процесса для того,

чтобы права доступа у создаваемого FIFO точно соответствовали

параметру вызова mknod() \*/

(void)umask(0);

/\* Попытаемся создать FIFO с именем aaa.fifo в текущей

директории \*/

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

/\* Если создать FIFO не удалось, печатаем об этом

сообщение и прекращаем работу \*/

printf("Can\'t create FIFO\n");

return 0;

}

if((result = fork()) < 0){

/\* Если создать процесс не удалось, сообщаем об этом и

завершаем работу \*/

printf("Can\'t fork child\n");

return 0;

} else if (result > 0) {

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

/\* Если открыть FIFO не удалось, печатаем об этом

сообщение и прекращаем работу \*/

printf("Can\'t open FIFO for writing\n");

return 0;

}

/\* Пробуем записать в FIFO 14 байт, т.е. всю строку

"Hello, world!" вместе с признаком конца строки \*/

size = write(fd, "Hello, world!", 14);

if(size != 14){

/\* Если записалось меньшее количество байт,то сообщаем

об ошибке и завершаем работу \*/

printf("Can\'t write all string to FIFO\n");

return 0;

}

/\* Закрываем входной поток данных и на этом родитель

прекращает работу \*/

close(fd);

printf("Parent exit\n");

} else {

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

/\* Если открыть FIFO не удалось, печатаем об этом

сообщение и прекращаем работу \*/

printf("Can\'t open FIFO for reading\n");

return 0;

}

/\* Пробуем прочитать из FIFO 14 байт в массив, т.е.

всю записанную строку \*/

size = read(fd, resstring, 14);

if(size < 0){

printf("Can\'t read string\n");

return 0;

}

/\* Печатаем прочитанную строку \*/

printf("%s\n",resstring);

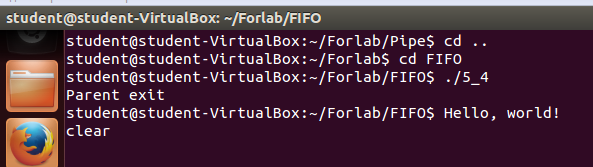
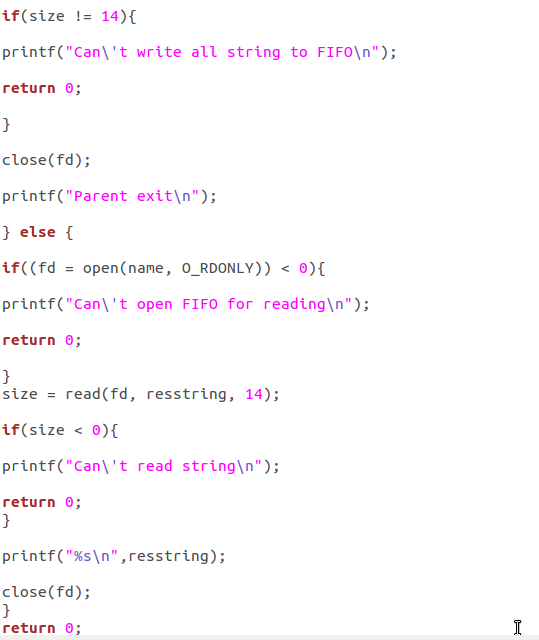
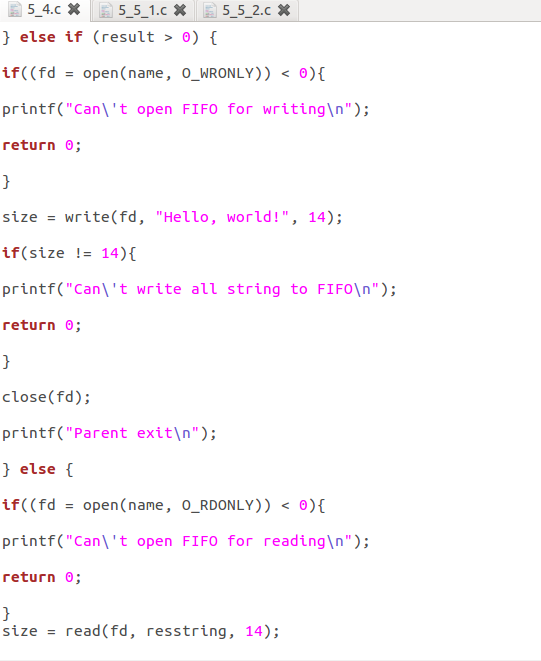
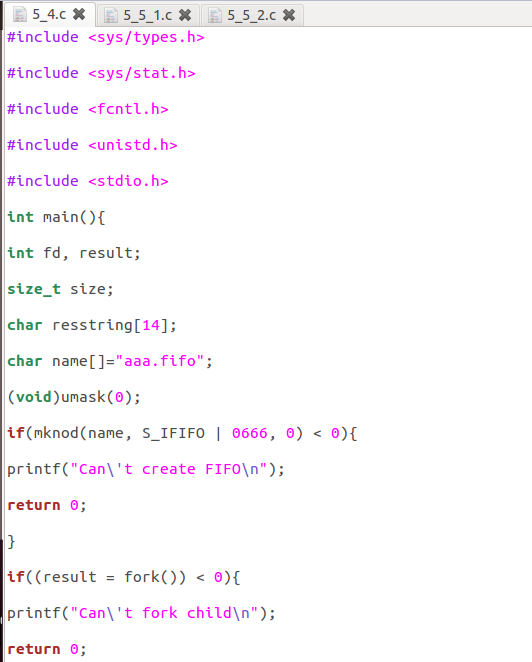
/\* Закрываем входной поток и завершаем работу \*/

close(fd);

}

return 0;

}



**Задание 13:** напишите на базе предыдущего примера две программы, одна из которых пишет информацию в *FIFO,* а вторая - читает из него, так чтобы между ними не было ярко выраженных родственных связей (то есть, чтобы ни одна из них не была потомком другой).

