Приложение 1. Отчет по лабораторной работе

**Федеральное государственное автономное образовательное учреждение высшего профессионального образования   
"Национальный исследовательский университет   
"Высшая школа экономики"**

  Факультет компьютерных наук

Департамент программной инженерии

Отчёт по лабораторной работе

**Лабораторная работа №5. (См. Семинар 5)**

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

по курсу «Операционные системы»

Выполнил:

Принял

Фомин Сергей Сергеевич

Оценка:

Москва 2020г.

Оглавление

[Задание 1 3](#_Toc38310856)

[Задание 2 3](#_Toc38310857)

[Задание 3 3](#_Toc38310858)

[Задание 4 4](#_Toc38310859)

[Задание 5 4](#_Toc38310860)

[Задание 6 4](#_Toc38310861)

[Задание 7 5](#_Toc38310862)

[Задание 8 5](#_Toc38310863)

[Задание 9 5](#_Toc38310864)

[Задание 10 5](#_Toc38310865)

**Для каждого задания указывается:**

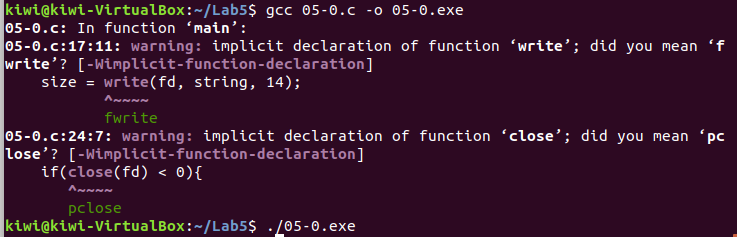
1. Формулировка задания.
2. Исходный текст программы.
3. Формат вызова программы.
4. Результат работы программы.
5. Объяснения полученных результатов.

# Задание 1

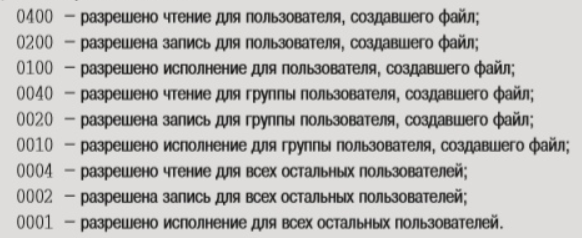
1. Откомпилируйте программу 05-0.c (в учебнике отсутствует, есть в архиве) и запустите ее на исполнение. Обратите внимание на несоответствие прав доступа у созданного файла правам, указанным в параметрах системного вызова open(). Объясните причину расхождений.

(3 балла)

Исходный код: <https://pastebin.com/a4vxCknM>

Формат вызова: Компиляция и запуск программы

Результат работы: Создание файла myfile со следующими правами доступа:

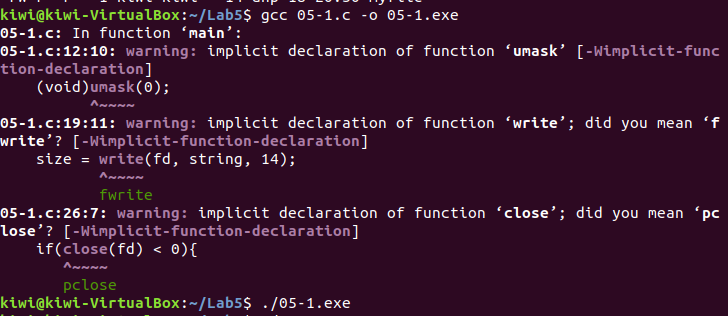
Объяснение: При создании файла реально устанавливаемые права доступа получаются из стандартной комбинации параметра mode и маски создания файлов текущего процесса umask, то есть они равны 0666 & ~O\_WRONLY. Параметр mode (0666) является обязательным, поскольку среди заданных флагов присутствует флаг O\_CREAT. Он задаётся как сумма восьмеричных значений:

# Задание 2

2. Откомпилируйте программу 05-1.c и запустите ее на исполнение. Обратите внимание на использование системного вызова umask() с параметром 0 для того, чтобы права доступа к созданному файлу точно соответствовали указанным в системном вызове open().

(3 балла)

Исходный код: <https://pastebin.com/yMGzSSdy>

Формат вызова: Компиляция и запуск программы

Результат работы: Создание файла myfile1 со следующими правами доступа:

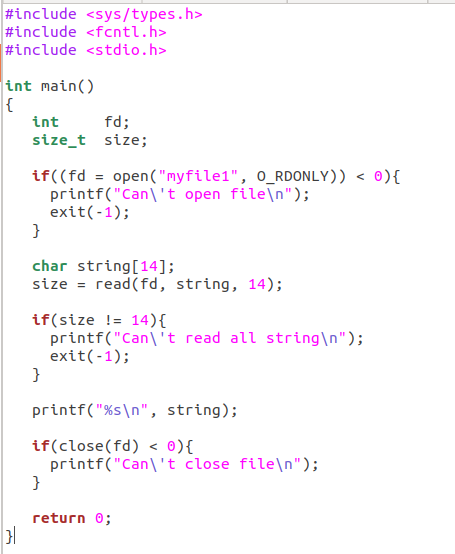
Объяснение:  
Разница между правами доступа файлов:

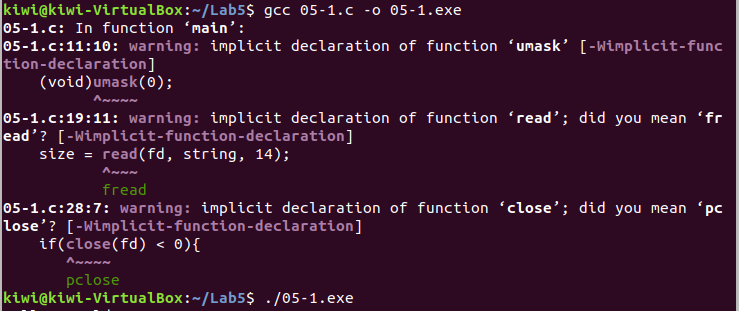
Причина:  
Функция umask() обнуляет маску создания файлов текущего процесса для того, чтобы права доступа у создаваемого файла точно соответствовали параметру вызова open().

# Задание 3

3. Измените программу 05-1.c так, чтобы она читала записанную ранее в файл информацию и выводила ее на экран. Все лишние операторы желательно удалить, оставив нужные.

(5 баллов – за правильное решение,  
при наличии лишних операторов — -1 балл за каждый,  
при удалении нужного — -1 за каждое лишнее удаление).

Исходный код (уже изменённой программы):

Формат вызова: Компиляция и запуск программы

Результат работы: программа читает записанную ранее в файл информацию и выводит ее на экран.

Объяснение: был удалён оператор umask(0), использовавшийся для обнуления маски создания файлов.

# Задание 4

4. Откомпилируйте программу 05-2.c (работа с **pipe** в рамках одного процесса), запустите на исполнение. Объясните результаты.

(5 баллов)

Исходный код: <https://pastebin.com/nXa0GLGc>

Формат вызова: Компиляция и запуск программы

Результат работы: в результате работы программы создаётся pipe, в который записывается строка, после чего она выводится на экран:

Объяснение: pipe представляет собой область памяти, недоступную пользовательским процессам напрямую, зачастую организованную в виде кольцевого буфера (хотя существуют и другие виды организации). По буферу при операциях чтения и записи перемещаются два указателя, соответствующие входному и выходному потокам(fd[2]).Выходной указатель никогда не может перегнать входной и наоборот (то есть ). Для создания нового экземпляра такого кольцевого буфера внутри операционной системы используется системный вызов pipe().

# Задание 5

5. Откомпилируйте программу 05-3.с (однонаправленная связь между предком и потомком), запустите на исполнение. Объясните результаты.

(5 баллов)

Исходный код: <https://pastebin.com/41mHY6BH>

Формат вызова: Компиляция и запуск программы

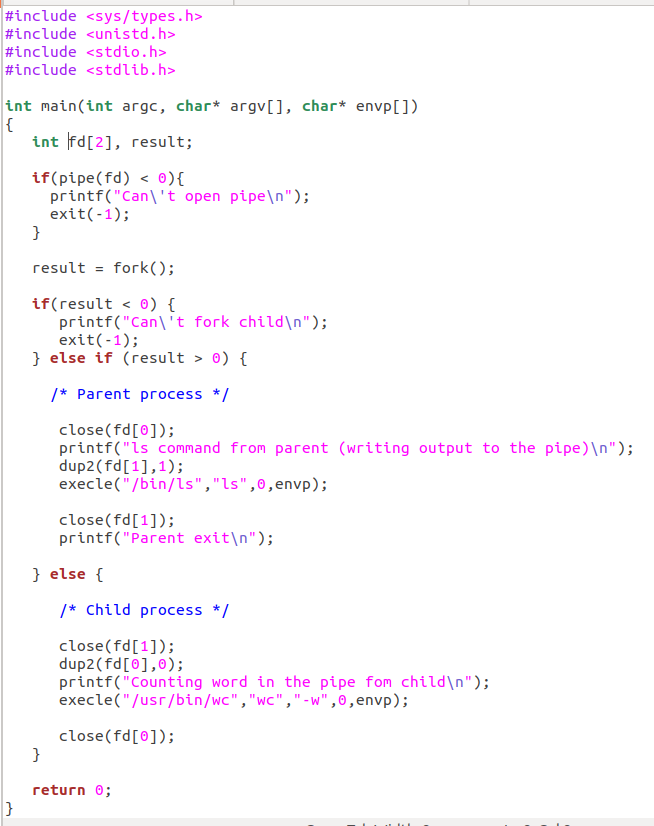
Результат работы: в результате работы программы создаётся дочерний процесс, который записывает строку в pipe, после чего она считывается дочерним процессом и выводится на экран

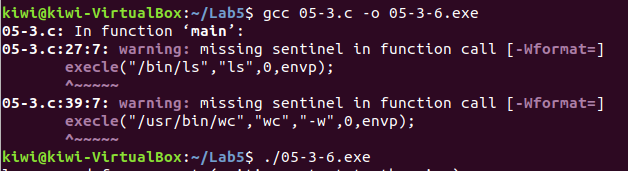
Объяснение: таблица открытых файлов наследуется дочерним процессом при порождении нового процесса системным вызовом fork(). Это позволяет организовать передачу информации через pipe между родственными процессами, имеющими общего прародителя, создавшего pipe.

# Задание 6

6. **Задача повышенной сложности**: модифицируйте пример 05-3.с для связи между собой двух родственных процессов, исполняющих **разные программы**.

Первому, кто предложил и реализовал новый для группы вариант, – 15 баллов, повторение уже предложенной идеи – 8 баллов.

Исходный код:

Формат вызова: Компиляция и запуск программы

  
Результат работы:

Объяснение: в результате работы программы создаётся дочерний процесс, который записывает список всех файлов находящихся в том же каталоге, что и исполняемый в pipe, после чего информация считывается дочерним процессом, в котором запускается программа wс параметром -w, которая выводит количество слов в pipe на экран.

# Задание 7

7. Модифицируйте программу 05-3.с для организации **двусторонней** связи, откомпилируйте ее и запустите на исполнение. Объясните результаты.

(7 баллов)

Исходный код:

# Задание 8

8. **Задача повышенной сложности**: определите размер pipe для вашей операционной системы.

(Оригинальная идея - 15 баллов).

(Повтор идеи и объяснение происходящего — 8 баллов).

Исходный код:

# Задание 9

9. Откомпилируйте программу 05-4.c, запустите её на исполнение. В этой программе информацией через **именованный программный канал** обмениваются процесс-родитель и процесс-ребенок.

(3 балла)

Исходный код: <https://pastebin.com/YCwsh7SD>

Формат вызова: Компиляция и запуск программы

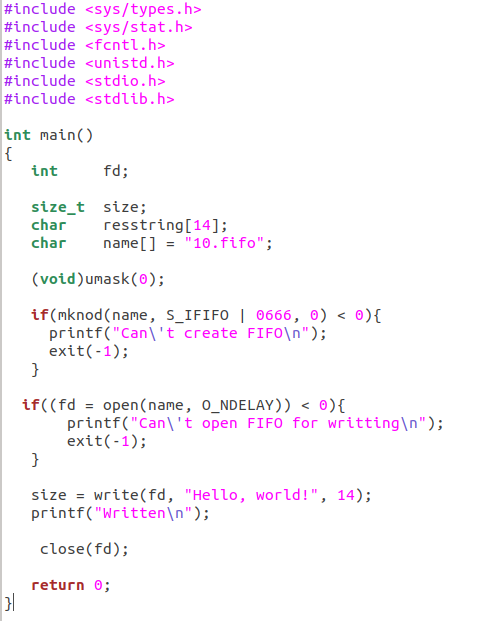
Результат работы: в результате работы программы между собой обмениваются информацией процесс-родитель и процесс-ребенок. Повторный запуск этой программы приведет к ошибке при попытке создания FIFO, так как файл с заданным именем уже существует.

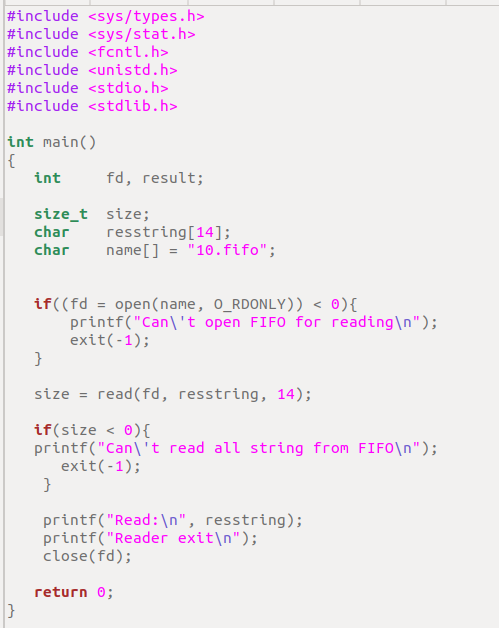
Объяснение: В отличие от pipe, данные о расположении FIFO в адресном пространстве ядра и его состоянии процессы могут получать не через родственные связи, а через файловую систему. Для этого при создании именованного pip’а на диске заводится файл специального типа (заводится файл-метка), обращаясь к которому процессы могут получить интересующую их информацию. Для создания FIFO используется системный вызов mknod(). После завершения процессов, работающих с FIFO, закрываются все ресурсы, использованные им, непрочитанная информация теряется. Однако остаётся файл-метка, который может использоваться для организации FIFO в дальнейшем.

# Задание 10

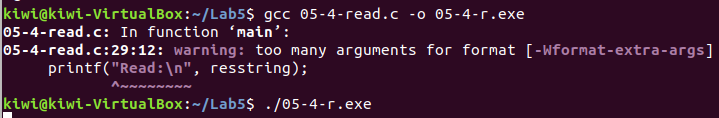
Напишите на базе предыдущего примера две **не** родственные программы, одна из которых пишет информацию в FIFO, а вторая – читает из него.

(7 баллов)

Исходный код: запись

чтение

Формат вызова: Компиляция и запуск программы



Результат работы: Первая программа создала требуемый файл, вторая – считала информацию и заблокировала процесс до тех пор, пока он вновь не откроется на запись

Объяснение: . Если FIFO открывается только для чтения, и флаг O\_NDELAY не задан, то процесс, осуществивший системный вызов, блокируется до тех пор, пока какой-либо другой процесс не откроет FIFO на запись. Если флаг O\_NDELAY задан, то возвращается значение файлового дескриптора, ассоциированного с FIFO. Если FIFO открывается только для записи, и флаг O\_NDELAY не задан, то процесс, осуществивший системный вызов, блокируется до тех пор, пока какой-либо другой процесс не откроет FIFO на чтение. Если флаг O\_NDELAY задан, то констатируется возникновение ошибки и возвращается значение -1. Задание флага O\_NDELAY в параметрах системного вызова open() приводит и к тому, что процессу, открывшему FIFO, запрещается блокировка при выполнении последующих операций чтения из этого потока данных и записи в него.