Именованные каналы

Отчёт по лабораторной работе №15

Мурашко В.В.

Содержание

# Теоретическое введение

С помощью труб могут общаться только родственные друг другу процессы, полученные с помощью fork(). Именованные каналы FIFO позволяют обмениваться данными с абсолютно ``чужим’’ процессом.

С точки зрения ядра ОС FIFO является одним из вариантов реализации трубы. Системный вызов mkfifo() предоставляет процессу именованную трубу в виде объекта файловой системы. Как и для любого другого объекта, необходимо предоставлять процессам права доступа в FIFO, чтобы определить, кто может писать, и кто может читать данные. Несколько процессов могут записывать или читать FIFO одновременно. Режим работы с FIFO - полудуплексный, т.е. процессы могут общаться в одном из направлений. Типичное применение FIFO - разработка приложений ``клиент - сервер’’.

# Цель работы

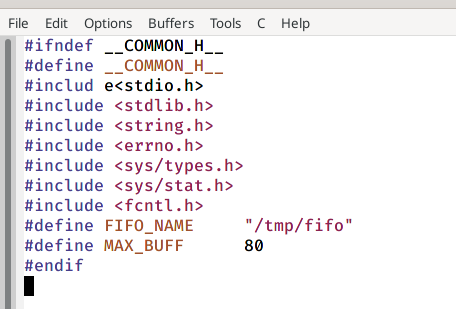
Приобретение практических навыков работы с именованными каналами.

# Задание

Изучите приведённые в тексте программы server.c и client.c. Взяв данные примеры за образец, напишите аналогичные программы, внеся следующие изменения: 1. Работает не 1 клиент, а несколько (например, два). 2. Клиенты передают текущее время с некоторой периодичностью (например, раз в пять секунд). Используйте функцию sleep() для приостановки работы клиента. 3. Сервер работает не бесконечно, а прекращает работу через некоторое время (например, 30 сек). Используйте функцию clock() для определения времени работы сервера. Что будет в случае, если сервер завершит работу, не закрыв канал?

# Выполнение лабораторной работы

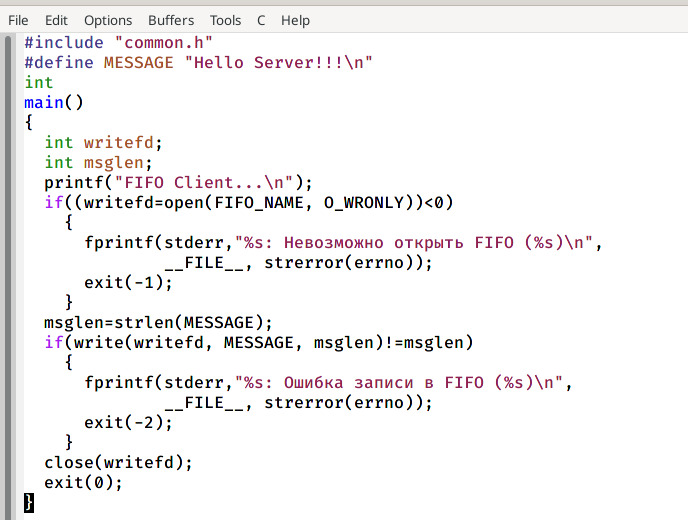
1. Я изучила приведённые в тексте программы server.c и client.c.



common.h



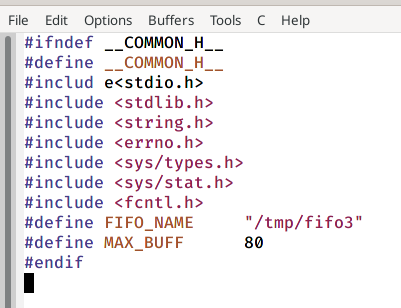
server.c



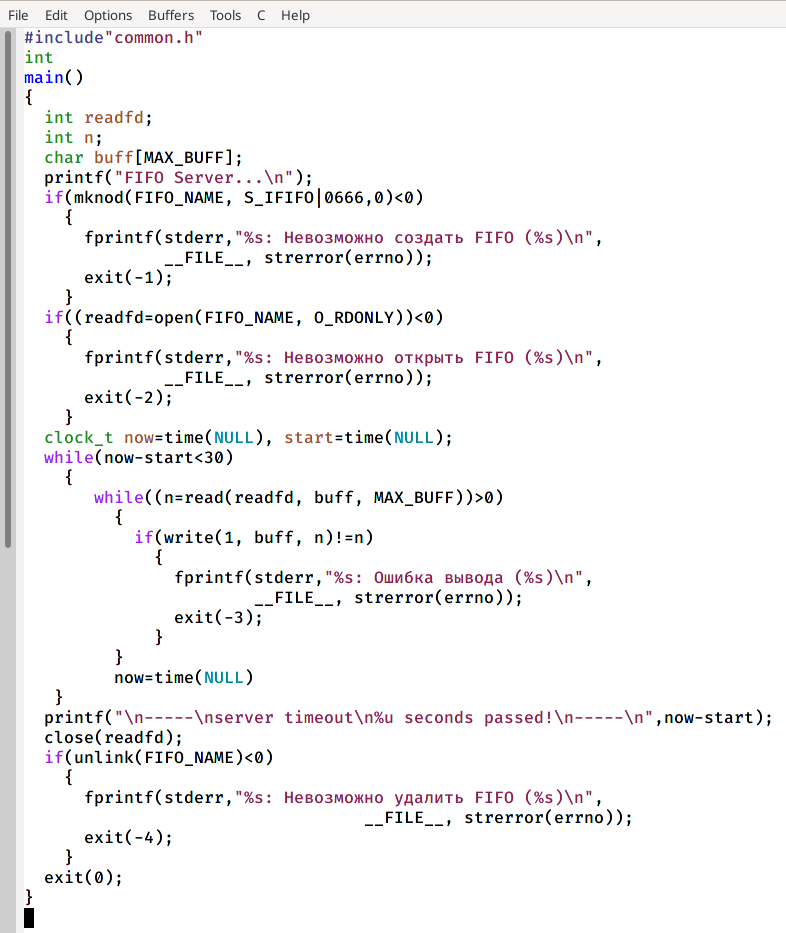
client.c

1. Я написала аналогичные программы, внеся следующие изменения:

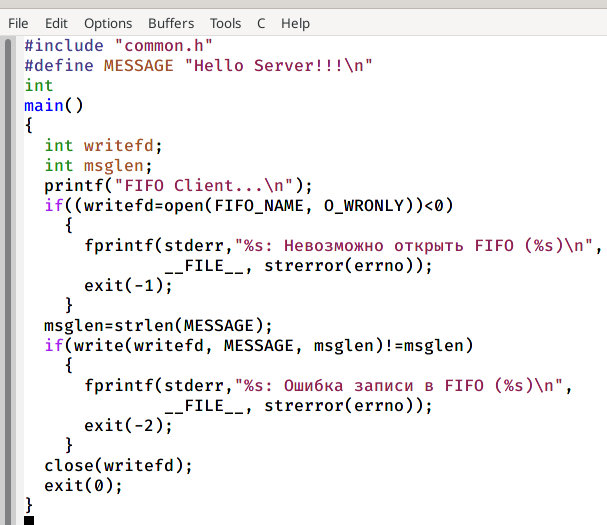
* работает не 1 клиент, а несколько (например, два).
* клиенты передают текущее время с некоторой периодичностью (например, раз в пять секунд). Использовала функцию sleep() для приостановки работы клиента.
* сервер работает не бесконечно, а прекращает работу через некоторое время (например, 30 сек). Использовала функцию clock() для определения времени работы сервера.



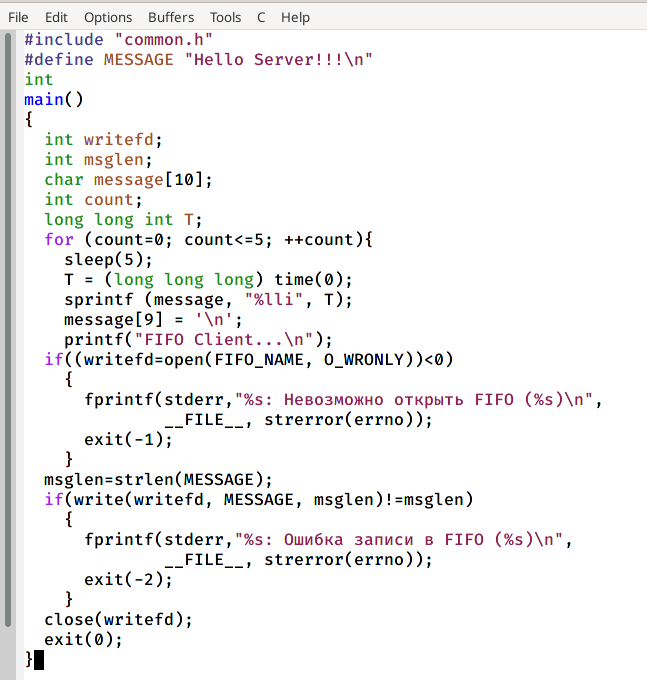
common.h



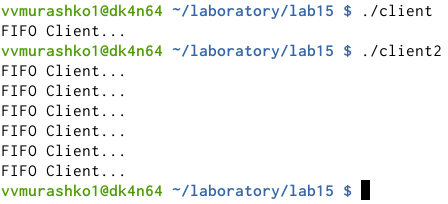
server.c



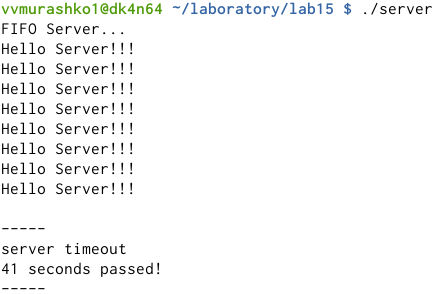
client.c



client2.c



./client.c и ./client2.c



./server

В случае, если сервер завершит работу, не закрыв канал, файл FIFO не удалится, поэтому его в следующий раз создать будет нельзя и вылезет ошибка, следовательно, работать ничего не будет.

# Выводы

Я приобрела практические навыки работы с именованными каналами.

# Библиография

https://www.opennet.ru/docs/RUS/linux\_parallel/node17.html

# Контрольные вопросы

1. Именованные каналы отличаются от неименованных наличием идентификатора канала, который представлен как специальный файл (соответственно имя именованного канала — это имя файла). Поскольку файл находится на локальной файловой системе, данное IPC используется внутри одной системы.
2. Создание неименованного канала из командной строки невозможно.
3. Создание именованного канала из командной строки возможно.
4. int read(int pipe\_fd, void *area, int cnt); int write(int pipe\_fd, void* area, int cnt); Первый аргумент этих вызовов - дескриптор канала, второй - указатель на область памяти, с которой происходит обмен, третий - количество байт. Оба вызова возвращают число переданных байт (или -1 - при ошибке).
5. int mkfifo (const char \*pathname, mode\_t mode) ; mkfifo(FIFO\_NAME, 0600) ; Первый параметр — имя файла, идентифицирующего канал, второй параметр маска прав доступа к файлу. Вызов функции mkfifo() создаёт файл канала (с именем, заданным макросом FIFO\_NAME).
6. При чтении меньшего числа байтов, чем находится в канале, возвращается требуемое число байтов, остаток сохраняется для последующих чтений. При чтении большего числа байтов, чем находится в канале или FIFO возвращается доступное число байтов.
7. При записи большего числа байтов, чем это позволяет канал или FIFO, вызов write(2) блокируется до освобождения требуемого места. При этом атомарность операции не гарантируется. Если процесс пытается записать данные в канал, не открытый ни одним процессом на чтение, процессу генерируется сигнал. Запись числа байтов, меньшего емкости канала или FIFO, гарантированно атомарно. Это означает, что в случае, когда несколько процессов одновременно записывают в канал, порции данных от этих процессов не перемешиваются.
8. В общем случае возможна много направленная работа процессов с каналом, т.е. возможна ситуация, когда с одним и тем же каналом взаимодействуют два и более процесса, и каждый из взаимодействующих каналов пишет и читает информацию в канал. Но традиционной схемой организации работы с каналом является однонаправленная организация, когда канал связывает два, в большинстве случаев, или несколько взаимодействующих процесса, каждый из которых может либо читать, либо писать в канал.
9. Write - Функция записывает length байтов из буфера buffer в файл, определенный дескриптором файла fd. Эта операция чисто ‘двоичная’ и без буферизации. Реализуется как непосредственный вызов DOS. С помощью функции write мы посылаем сообщение клиенту или серверу.
10. Строковая функция strerror - функция языков C/C++, транслирующая код ошибки, который обычно хранится в глобальной переменной errno, в сообщение об ошибке, понятном человеку. Ошибки эти возникают при вызове функций стандартных Си-библиотек. Возвращенный указатель ссылается на статическую строку с ошибкой, которая не должна быть изменена программой. Дальнейшие вызовы функции strerror перезапишут содержание этой строки. Интерпретированные сообщения об ошибках могут различаться, это зависит от платформы и компилятора.