Отчёт по лабораторной работе №14

Именованные каналы

Желдакова Виктория Алексеевна

Содержание

# 1 Цель работы

Приобретение практических навыков работы с именованными каналами.

# 2 Теоретическое введение

Канал - это средство связи стандартного вывода одного процесса со стандартным вводом другого. Каналы - это старейший из инструментов IPC, существующий приблизительно со времени появления самых ранних версий оперативной системы UNIX. Они предоставляют метод односторонних коммуникаций (отсюда термин half-duplex) между процессами.

Эта особенность широко используется даже в командной строке UNIX (в shell-е).

Именованные каналы во многом работают так же, как и обычные каналы, но все же имеют несколько заметных отличий.

Именованные каналы существуют в виде специального файла устройства в файловой системе. Процессы различного происхождения могут разделять данные через такой канал. Именованный канал остается в файловой системе для дальнейшего использования и после того, как весь ввод/вывод сделан.

# 3 Выполнение лабораторной работы

Изучили приведённые в тексте программы server.c и client.c. Взяв данные примеры за образец, написали аналогичные программы, (рис. 1, рис. 2 и рис. 3) внеся следующие изменения:

1. Работает не 1 клиент, а несколько.
2. Клиенты передают текущее время с некоторой периодичностью.
3. Сервер работает не бесконечно, а прекращает работу через некоторое время.

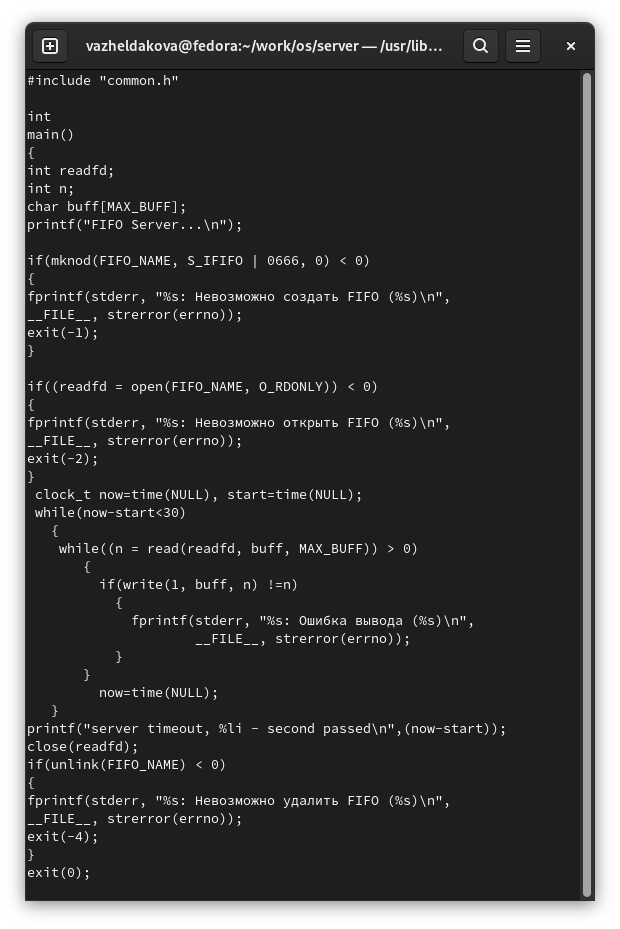


Рис. 1: Содержимое server.c

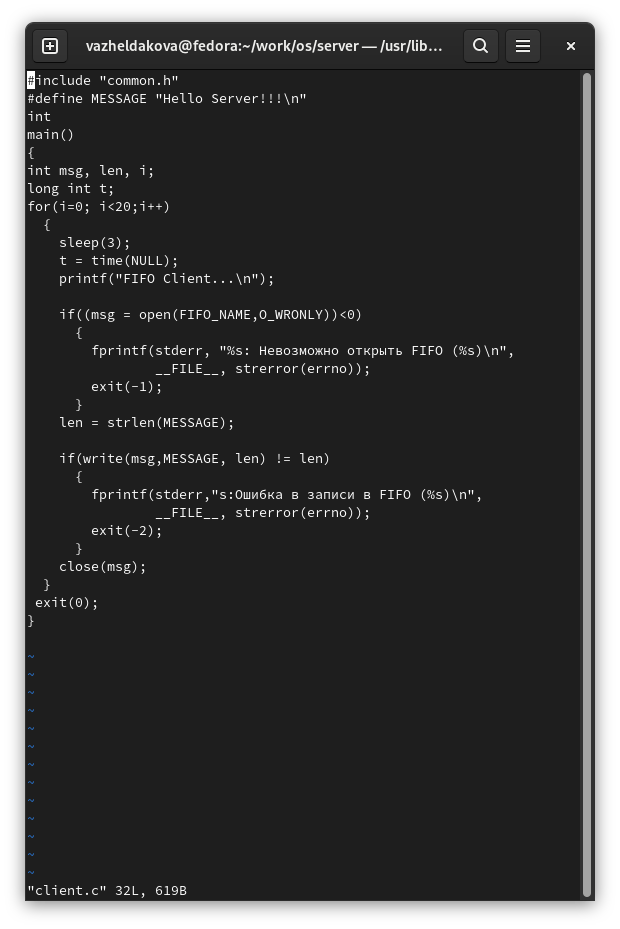


Рис. 2: Содержимое client.c

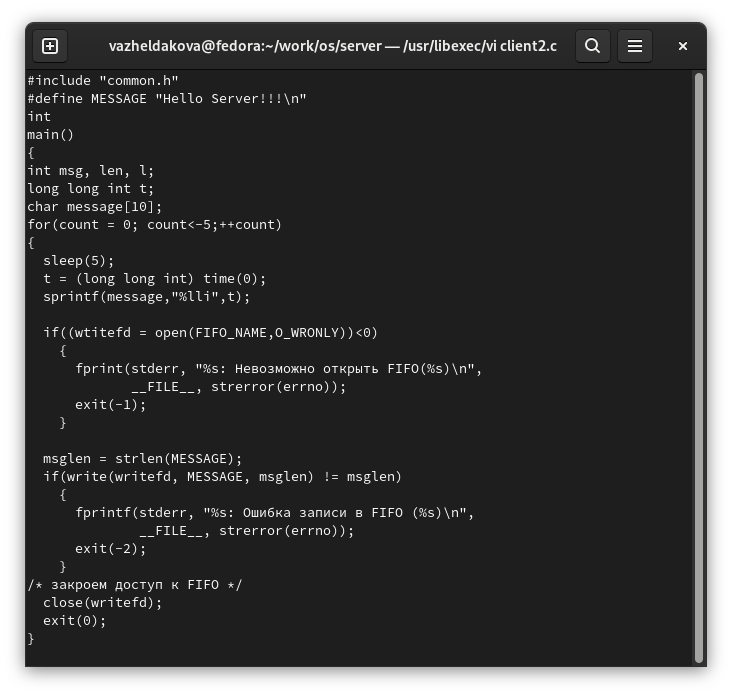


Рис. 3: Содержимое client2.c

Скопировали содержимое файлов common.h и Makefile из лабораторной работы (рис. 4 и рис. 5)

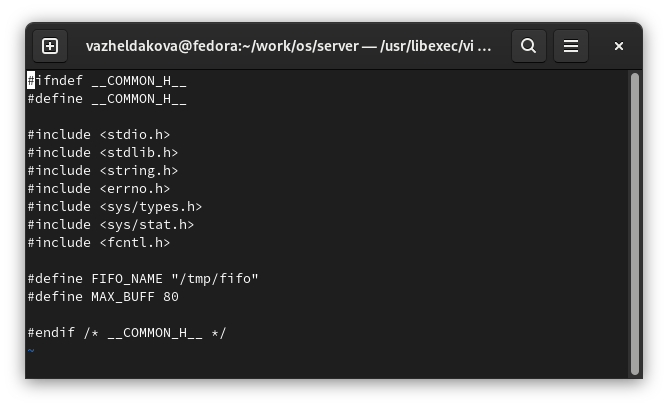


Рис. 4: Содержимое common.h

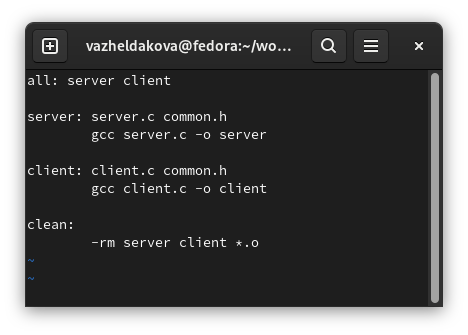


Рис. 5: Содержимое Makefile

Запустили make, server в одном окне и client в другом (рис. 6)



Рис. 6: Работа программы

Сообщения передевалась с периодичностью в несколько секунд, а по истечении 30 секунд - канал закрылся, и вывелось сообщение об этом.

Что будет в случае, если сервер завершит работу, не закрыв канал?

При завершении программы каналы автоматически закрываются.

# 4 Выводы

Приобрели практических навыков работы с именованными каналами.

# 5 Контрольные вопросы

1. В чем ключевое отличие именованных каналов от неименованных?

Именованные каналы отличаются от неименованных наличием идентификатора канала, который представлен как специальный файл (соответственно имя именованного канала — это имя файла).

1. Возможно ли создание неименованного канала из командной строки?

Создание неименованного канала из командной строки возможно командой pipe.

1. Возможно ли создание именованного канала из командной строки?

Создание именованного канала из командной строки возможно с помощью mkfifo.

1. Опишите функцию языка С, создающую неименованный канал.

Функция языка С, создающая неименованный канал: 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).

1. Опишите функцию языка С, создающую именованный канал.

Функция языка С, создающая именованный канал: int mkfifo (const char \*pathname, mode\_t mode); 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): mkfifo(FIFO\_NAME, 0600);

1. Что будет в случае прочтения из fifo меньшего числа байтов, чем находится в канале? Большего числа байтов?

При чтении меньшего числа байтов, возвращается требуемое число байтов, остаток сохраняется для следующих чтений. При чтении большего числа байтов, возвращается доступное число байтов.

1. Аналогично, что будет в случае записи в fifo меньшего числа байтов, чем позволяет буфер? Большего числа байтов?

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. 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 SIGPIPE, а вызов write(2) возвращает 0 с установкой ошибки (errno=ЕР1РЕ) (если процесс не установил обработки сигнала SIGPIPE, производится обработка по умолчанию – процесс завершается).

1. Могут ли два и более процессов читать или записывать в канал?

Два и более процессов могут читать и записывать в канал.

1. Опишите функцию write (тип возвращаемого значения, аргументы и логику работы). Что означает 1 (единица) в вызове этой функции в программе server.c (строка 42)?

Функция write записывает length байтов из буфера buffer в файл, определенный дескриптором файла fd. Эта операция чисто ‘двоичная’ и без буферизации. При единице возвращает действительное число байтов. Функция write возвращает число действительно записанных в файл байтов или -1 при ошибке, устанавливая при этом errno.

1. Опишите функцию strerror.

Строковая функция strerror - функция языков C/C++, транслирующая код ошибки, который обычно хранится в глобальной переменной errno, в сообщение об ошибке, понятном человеку.