Лабораторная работа №14.

Именованные каналы

Козлов Всеволод Павлович

Содержание

# 1 Цель работы

Приобретение практических навыков работы с именованными каналами.

# 2 Выполнение лабораторной работы

Создал файл client.c (рис. [[1](#fig:1)])

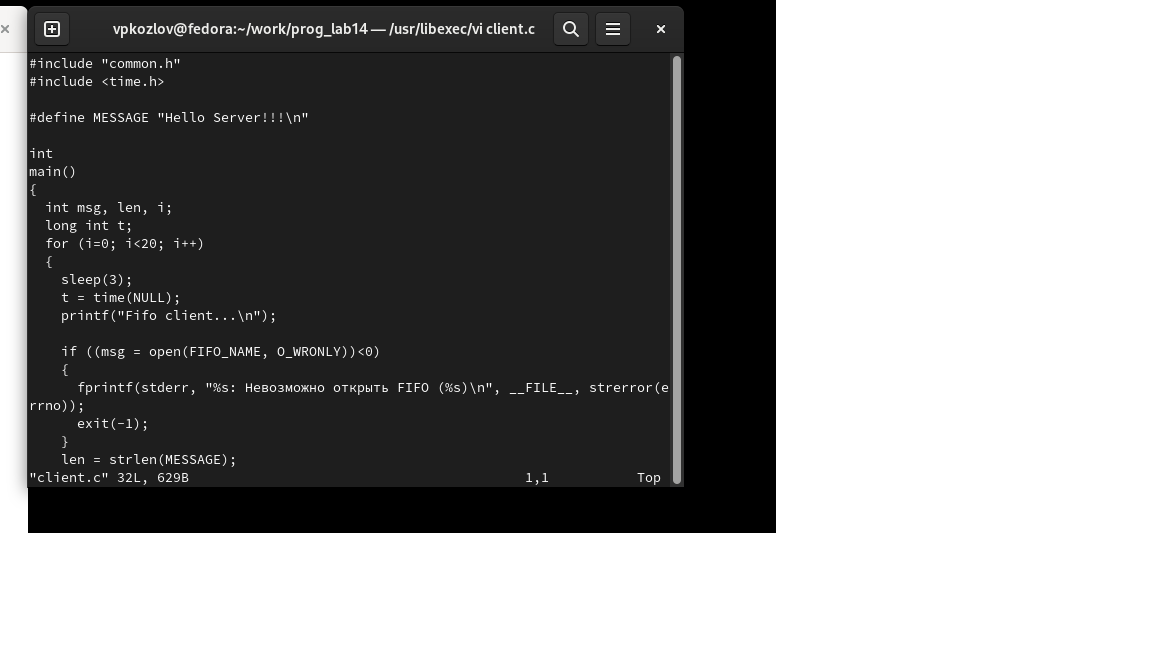


Figure 1: client.c

Создал client2.c (рис. [[2](#fig:2)])

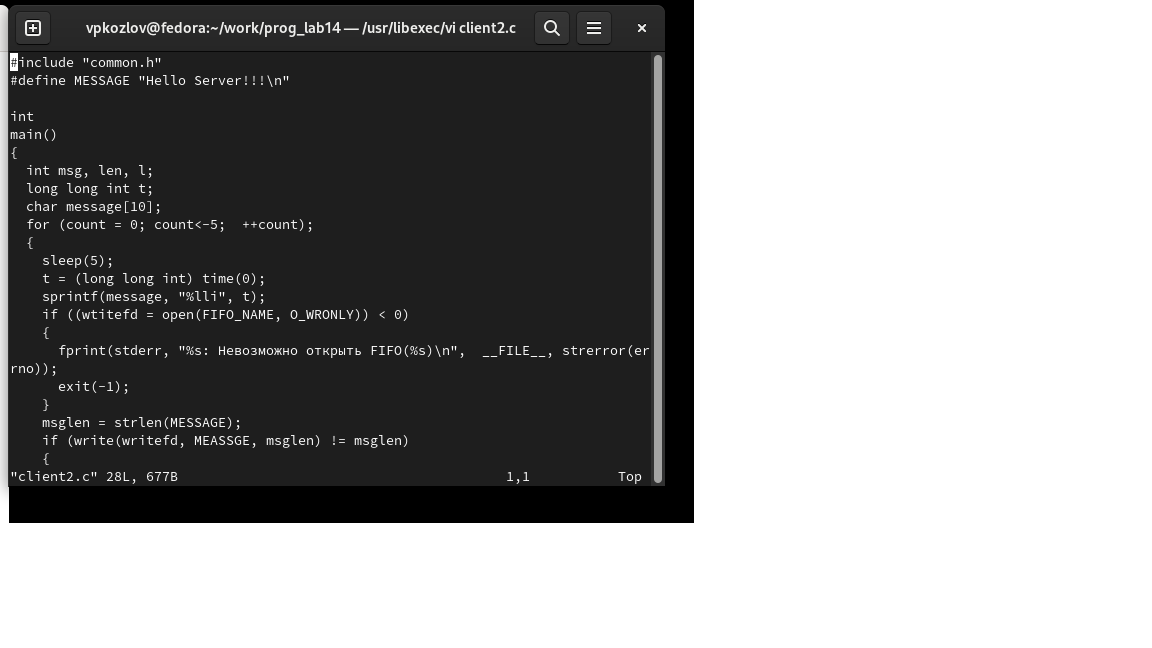


Figure 2: client2.c

Создал common.h (рис. [[3](#fig:3)])

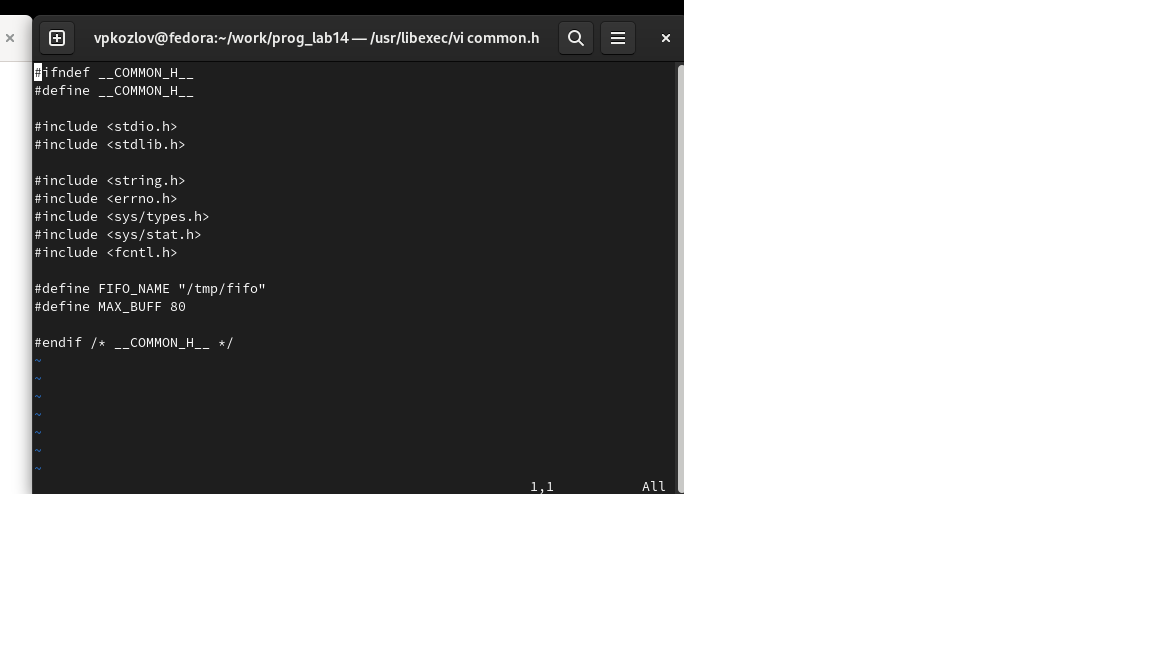


Figure 3: common.h

Создал server.c (рис. [[4](#fig:4)])

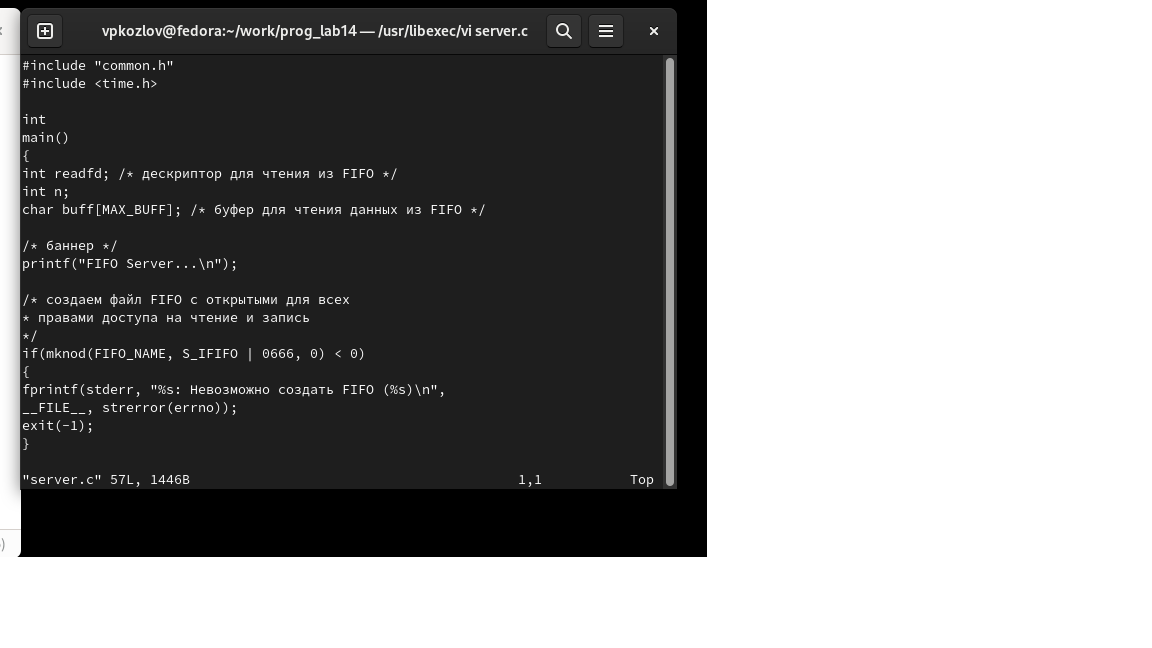


Figure 4: server.c

Создал Makefile (рис. [[5](#fig:5)])

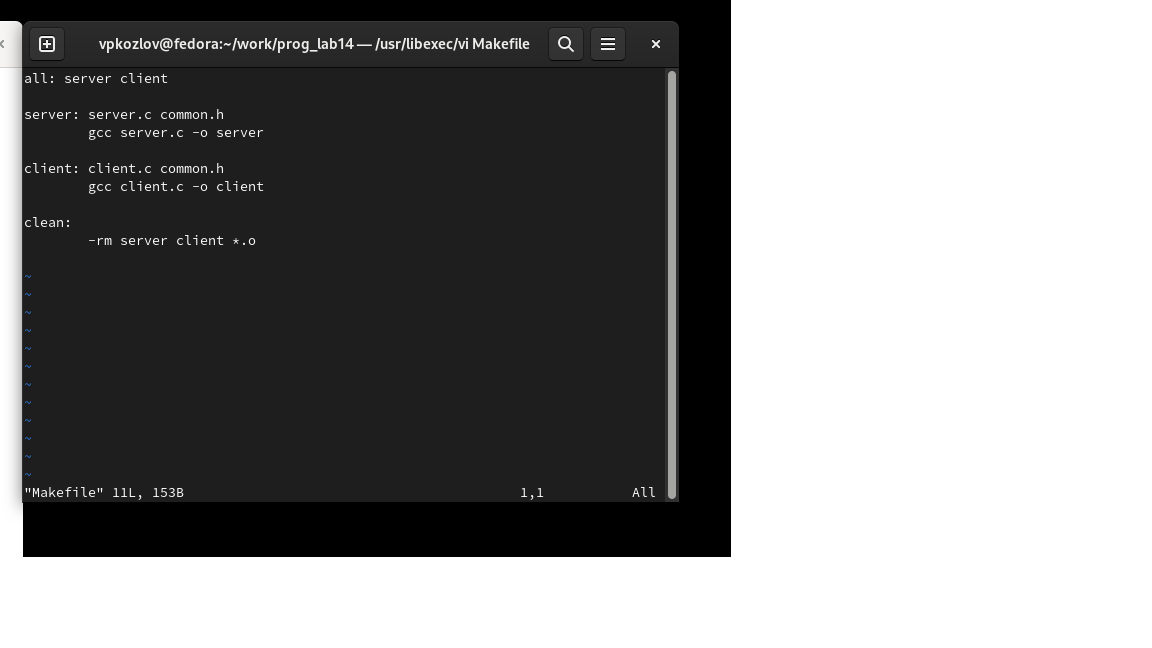


Figure 5: Makefile

Запустил команду make (рис. [[6](#fig:6)])

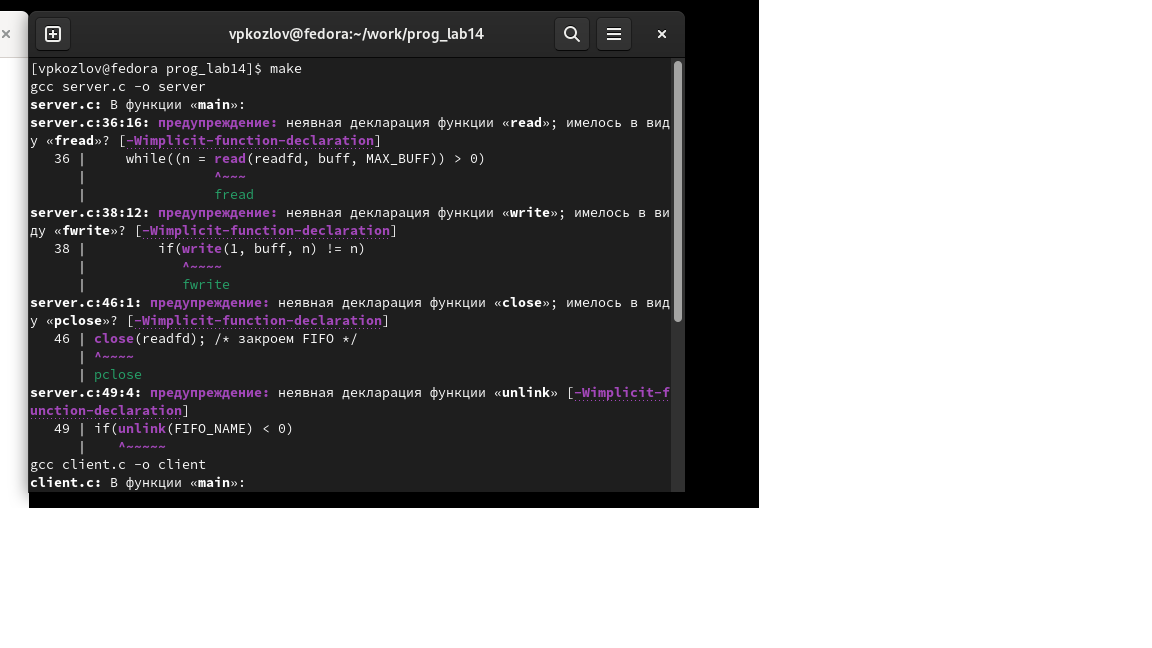


Figure 6: Запуск команды make

Запустил исполняемые файлы (рис. [[7](#fig:7)])

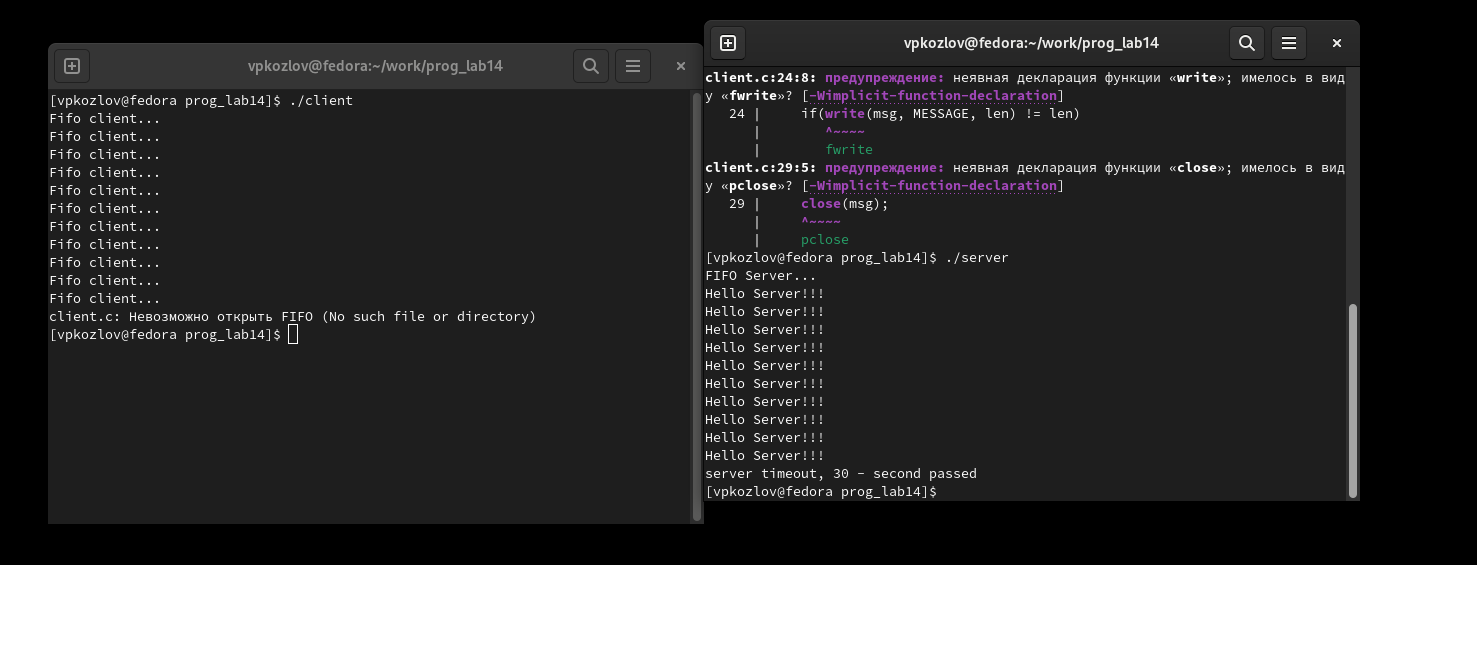


Figure 7: Запуск исполняемых файлов

# 3 Контрольные вопросы

1. Именованные каналы отличаются от неименованных наличием идентификатора канала, который представлен как специальный файл (соответственно имя именованного канала — это имя файла).
2. Создание неименованного канала из командной строки возможно командой pipe.
3. Создание именованного канала из командной строки возможно с помощью mkfifo.
4. 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).
5. 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);
6. При чтении меньшего числа байтов, возвращается требуемое число байтов, остаток сохраняется для следующих чтений. При чтении большего числа байтов, возвращается доступное число байтов 7. 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).
7. Два и более процессов могут читать и записывать в канал.
8. 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.
9. Строковая функция strerror - функция языков C/C++, транслирующая код ошибки, который обычно хранится в глобальной переменной errno, в сообщение об ошибке, понятном человеку.

# 4 Выводы

Приобрел практические навыки работы с именованными каналами.

# Список литературы