Лабораторная работа 15

Дисциплина: Операционные системы

Куликов Максим Игоревич

Содержание

# Цель работы

Приобретение практических навыков работы с именованными каналами.

# Задание

1. Ознакомиться с теоретическим материалом.
2. Выполнить работу.

# Выполнение лабораторной работы

1. Создаю в новом каталоге 4 скрипта (рис. -fig. 1)

Figure 1: Создание файлов

Figure 1: Создание файлов

1. Содержимое 4-х скриптов. В файл common.h добавил стандартные заголовочные файлы unistd.h и time.h, необходимые для работы кодов других файлов. Common.h предназначен для заголовочных файлов, чтобы в остальных программах их не прописывать каждый раз. В файл server.c добавил цикл while для контроля за временем работы сервера. Разница между текущим временем time(NULL) и временем начала работы clock\_t start=time(NULL) (инициализация до цикла) не должна превышать 30 секунд. В файл client.c добавил цикл, который отвечает за количество сообщений о текущем времени (4 сообщения). Makefile я оставил без изменения (рис. -fig. 2)

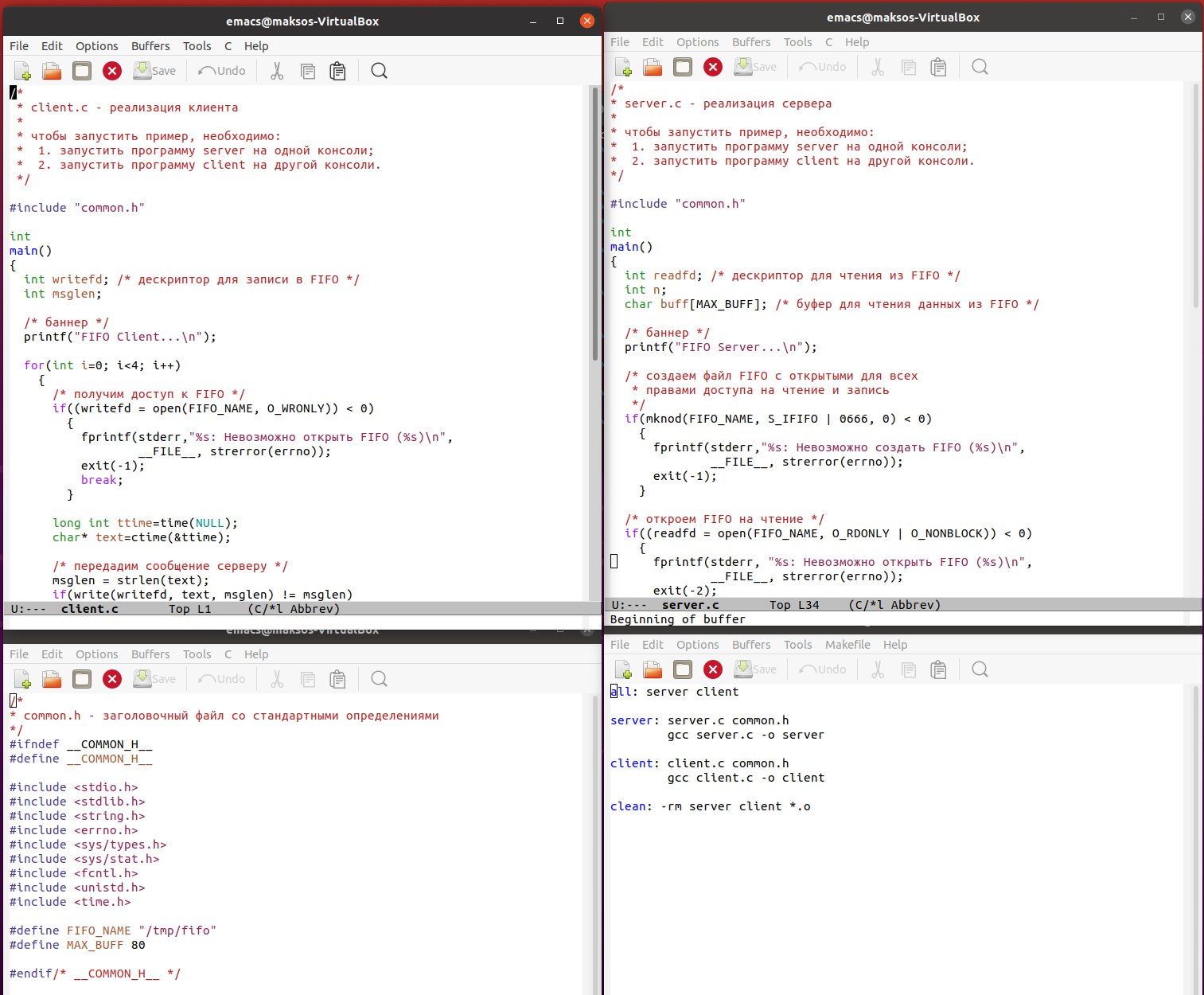


Figure 2: Скрипты

1. Выполняю компиляцию файлов с помощью Makefile. (рис. -fig. 3)

Figure 3: Компиляция

Figure 3: Компиляция

1. Запускаю сервер в 1 терминале. Запускаю 2 клиента из других терминалов. Они работают верно (каждый клиент выводит в терминале, где запустили сервер, по 4 сообщения) (рис. -fig. 4)

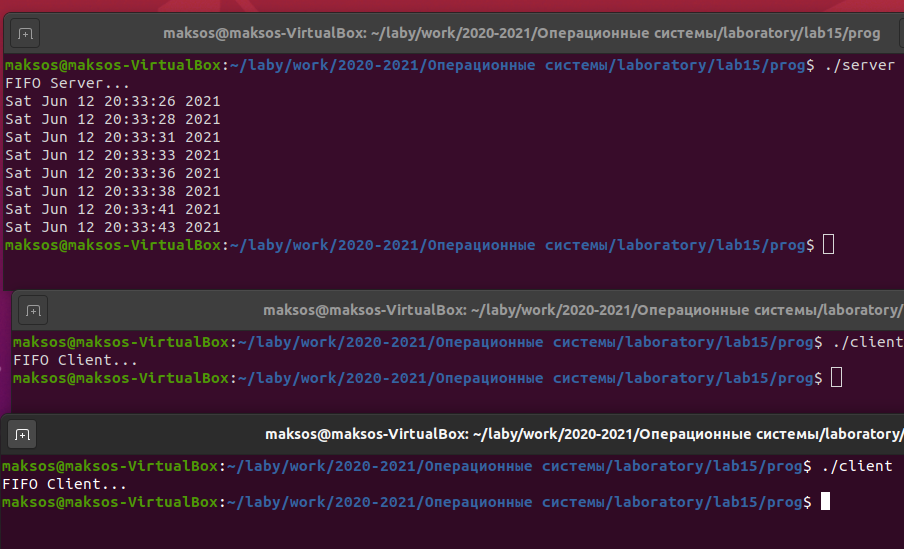


Figure 4: Проверка работы программы

1. Проверка работы сервера. Он должен закрываться автоматически через 30 секунд. Работает верно (рис. -fig. 5)

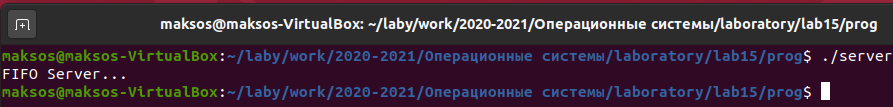


Figure 5: Проверка работы сервера

# Выводы

Приобрёл практические навыки работы с именованными каналами.

# Контрольные вопросы

1. Именованные каналы отличаются от неименованных наличием идентификатора канала, который представлен как специальный файл (соответственно имя именованного канала — это имя файла).
2. Для создания неименованного канала используется системный вызов pipe. Массив из двух целых чисел является выходным параметром этого системного вызова.
3. Вы можете создавать именованные каналы из командной строки и внутри программы. С давних времен программой создания их в командной строке была команда: mknod - $ mknod имя\_файла , однако команды mknod нет в списке команд X/Open, поэтому она включена не во все UNIX-подобные системы. Предпочтительнее применять в командной строке - $ mkfifo имя\_файла.
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).

1. 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);
2. 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.
3. 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.
4. В общем случае возможна многонаправленная 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.
5. 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.

1. 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.