# Презетация по лабораторной работе №15

### Средства, применяемые при разработке программного обеспечения в ОС типа UNIX/Linux

## Российский Университет Дружбы Народов

### Факультет Физико-Математических и Естественных Наук

*Дисциплина: Операционные системы*

Студент: Бен бадр Ясмин

Группа: НКНдб-01-20

2021г.

## Введение Именованные каналы

В программировании именованный канал или именованный конвейер — один из методов межпроцессного взаимодействия, расширение понятия конвейера в Unix и подобных ОС. Именованный канал позволяет различным процессам обмениваться данными, даже если программы, выполняющиеся в этих процессах, изначально не были написаны для взаимодействия с другими программами. Это понятие также существует и в Microsoft Windows, хотя там его семантика существенно отличается. Традиционный канал — «безымянен», потому что существует анонимно и только во время выполнения процесса. Именованный канал — существует в системе и после завершения процесса. Он должен быть «отсоединён» или удалён, когда уже не используется. Процессы обычно подсоединяются к каналу для осуществления взаимодействия между ними.

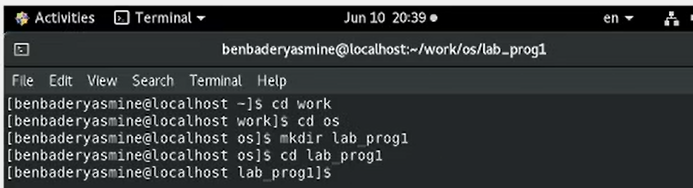
Вместо традиционного, безымянного конвейера оболочки (англ. shell pipeline), именованный канал создаётся явно с помощью mknod или mkfifo, и два различных процесса могут обратиться к нему по имени.

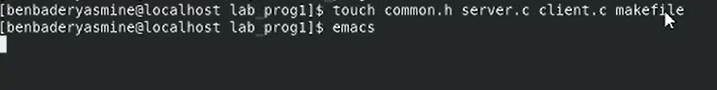
Например, можно создать канал и настроить gzip на сжатие того, что туда попадает:

mkfifo pipe gzip -9 -c < pipe > out Параллельно в другом процессе можно выполнить:

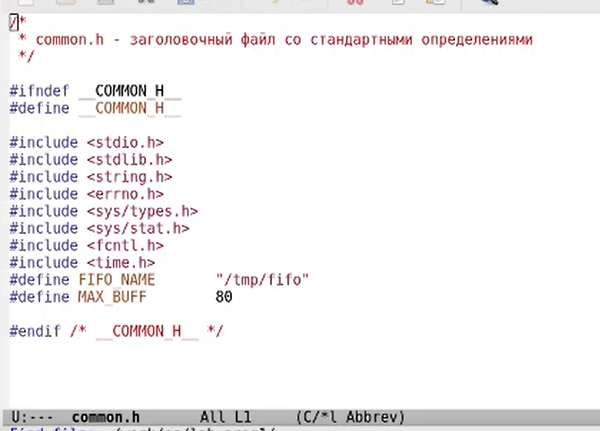
cat file > pipe ##резюме Программные каналы - мощнейший инструмент Юниксовидных операционных систем. Создание программных каналов многократно ускоряет процесс обработки данных, сокращает количество "писанины" в командной строке, а также позволяет получать результат в наиболее удобном виде. ## Цель работы Приобретение практических навыков работы с именованными каналами.

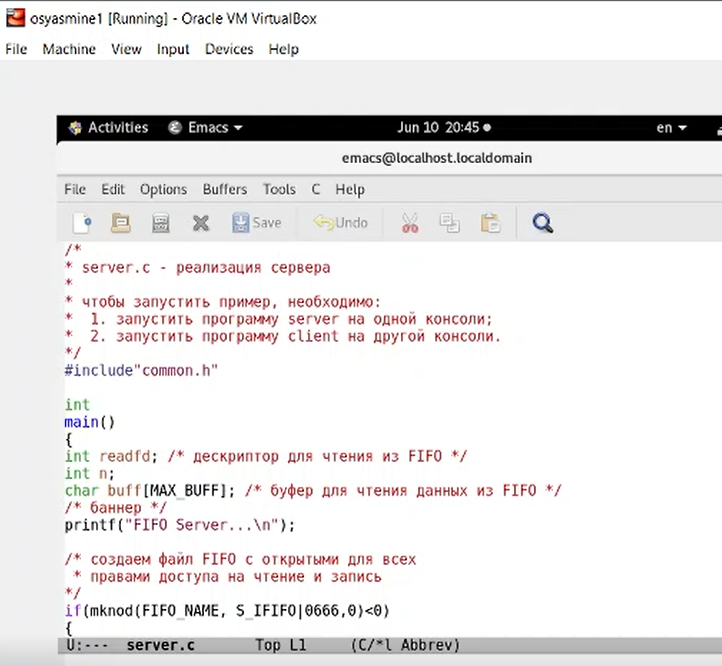
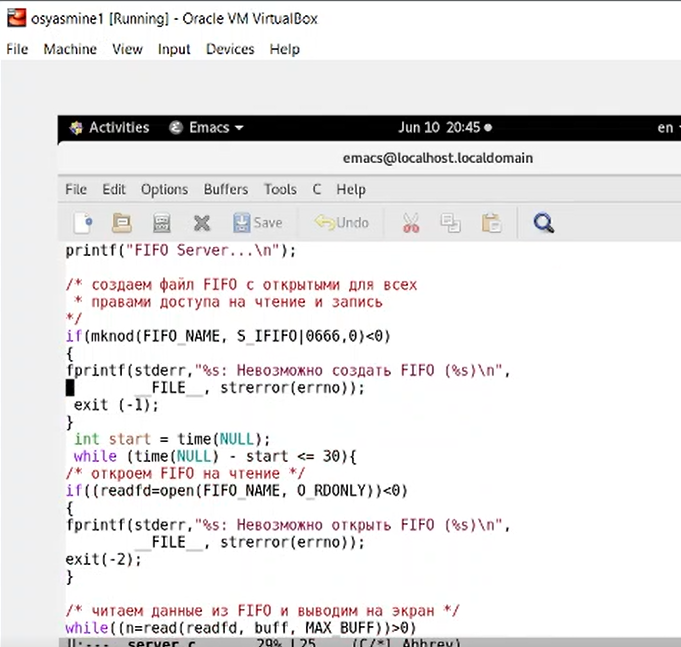
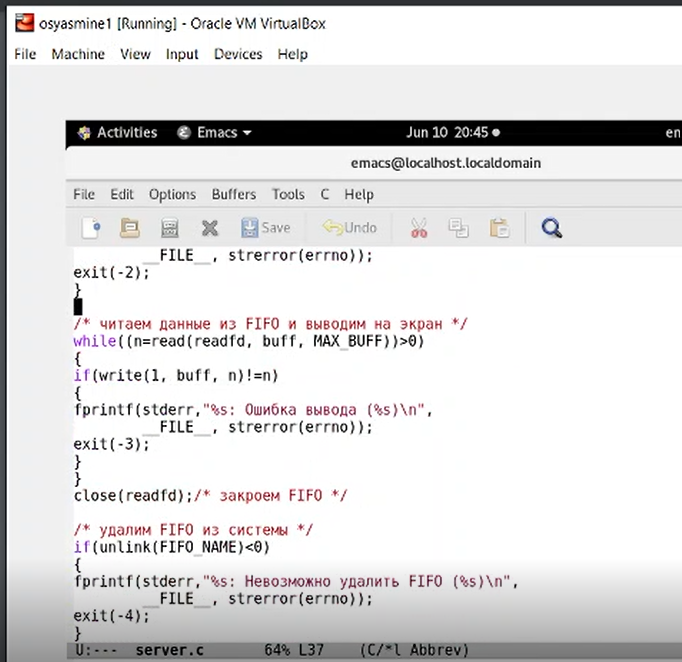
Ход работы 1. Перешла в рабочный каталогах *work/os/lab\_prog1* , чтобы начинать сделать лаб15

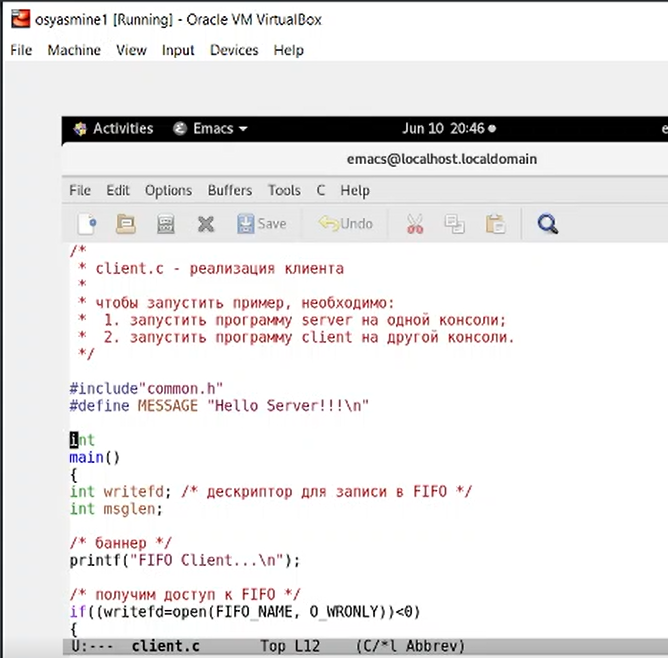
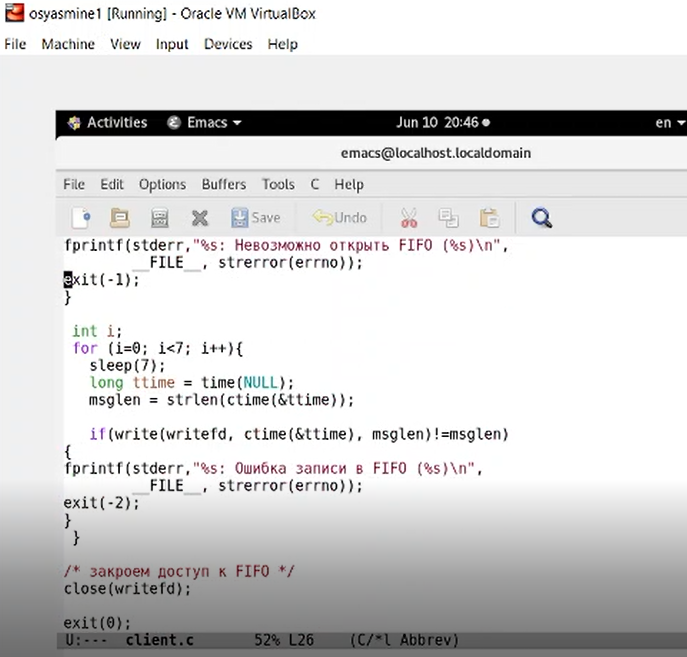


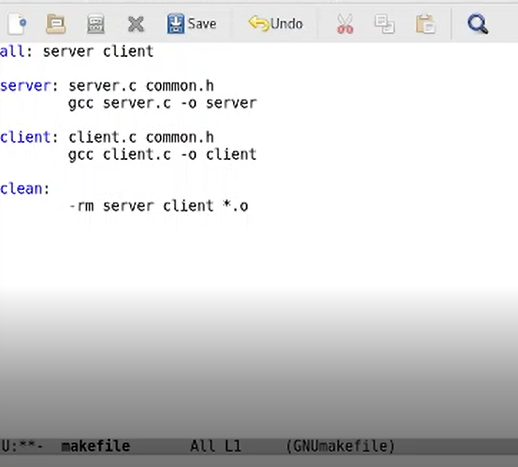
1. создала три файла и в каждом написала прграмму 

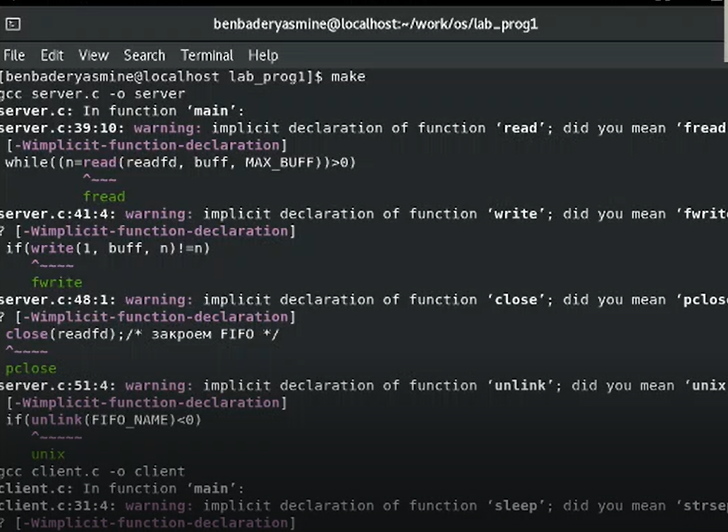
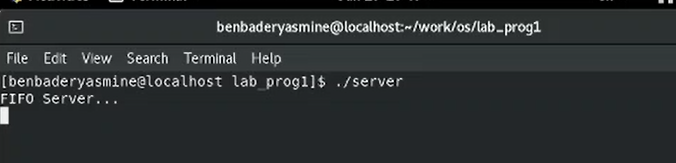
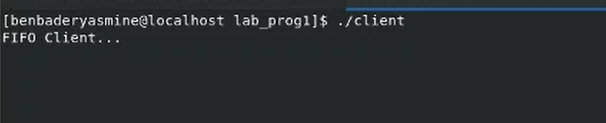
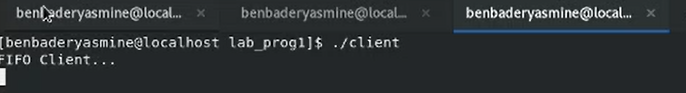
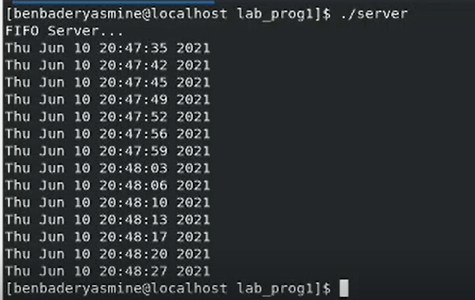
**Файл common.h**



**Файл server.c**   

**Файл client.c**  

**Файл makefile** 

1. разработалась в программах -примерах Сервер выключаеться после 3 сообщений в программе-примере сообщения принимались только определенной длинны равной  
      
2. получила результат 

## Вывод

В результате работы , я приобрела 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).
5. 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, 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 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.
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