Отчёт по лабораторной работе №14

Операционные системы

Никулина Ксения Ильинична

Содержание

# 1 Цель работы

Приобретение практических навыков работы с именованными каналами

# 2 Задание

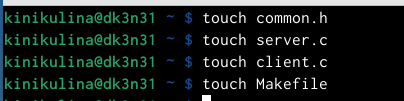
Изучить приведённые в тексте программы server.c и client.c. Взяв данные примеры за образец, напишите аналогичные программы, внеся следующие изменения:

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?

# 3 Выполнение лабораторной работы

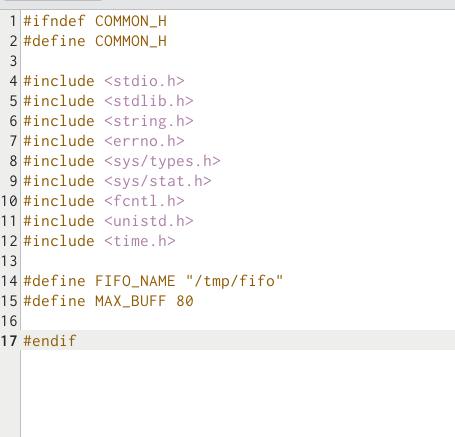
Для начала изучили материал лабораторной работы. Далее на основе примеров напишем аналогичные программы, но с изменениями.

1. Для начала создадим необходимые файлы для работы. (рис. ??).



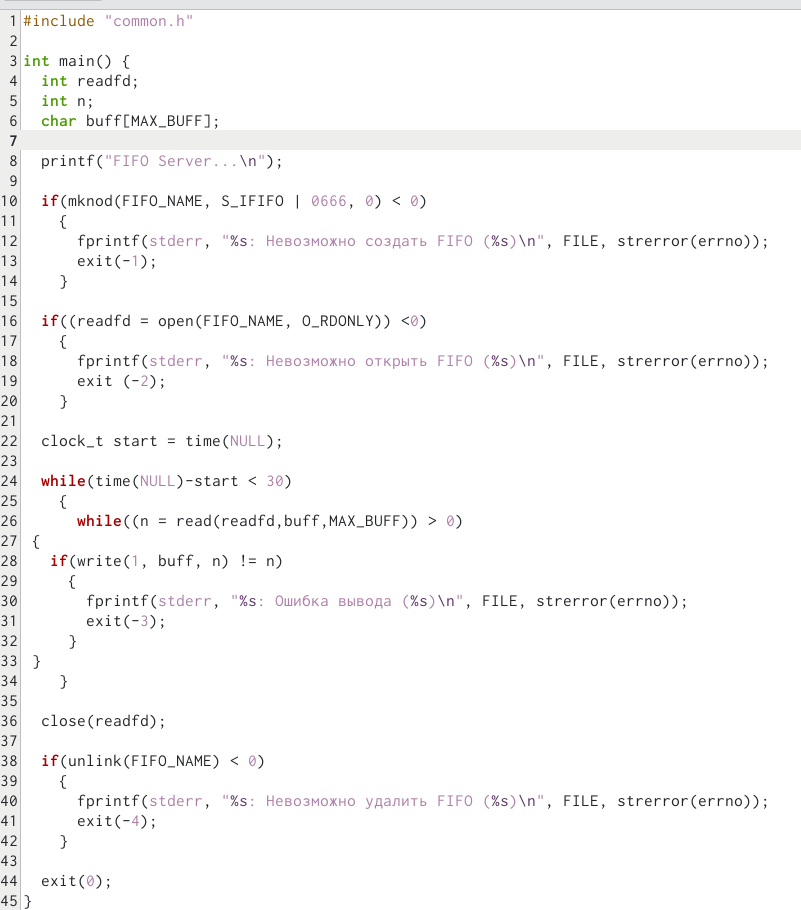
Создание файлов

Затем изменим коды программ, данных в лабораторной работе. В файл *common.h* добавим стандартные заголовочные файлы: “unistd.h”, “time.h”. Это необходимо для работы других файлов. Этот файл является заголовочным, чтобы в остальных програмах не прописывать одно и то же каждый раз.(рис. ??).



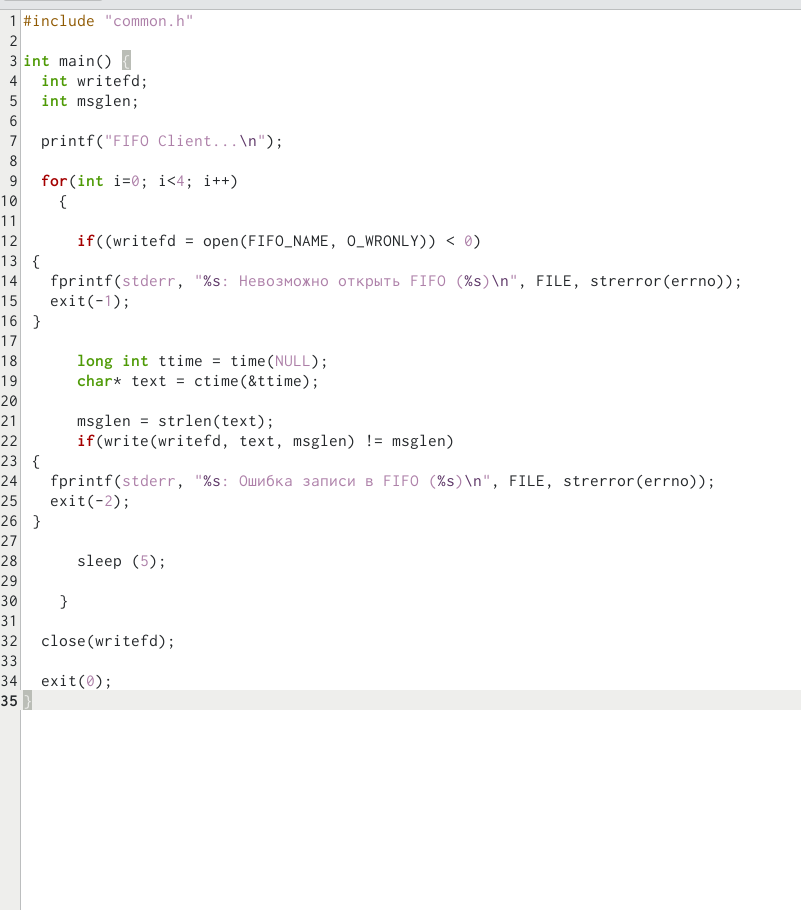
Файл “common.h”

1. Затем в файл *server.c* добавляем цикл “while” для контроля за верменем работы сервера. Разница между текущим временем и началом работы не должна превышать 30 секунд.(рис. ??),



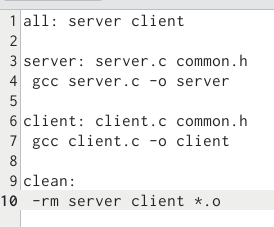
Файл “server.c”

1. В файл *client.c* добавим цикл, который отвечает за количество сообщений о текущем времени (4 сообщения). С промощью команды “sleep” приостановим работу клиента на 5 секунд. (рис. ??).



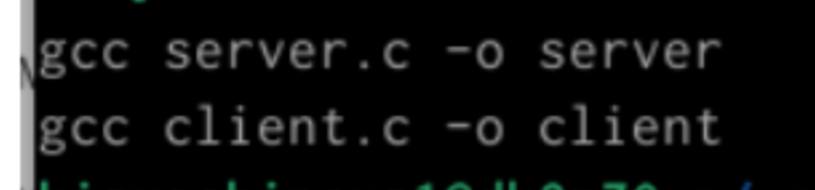
Файл “client.c”

Makefile оставили без изменений.(рис. ??).



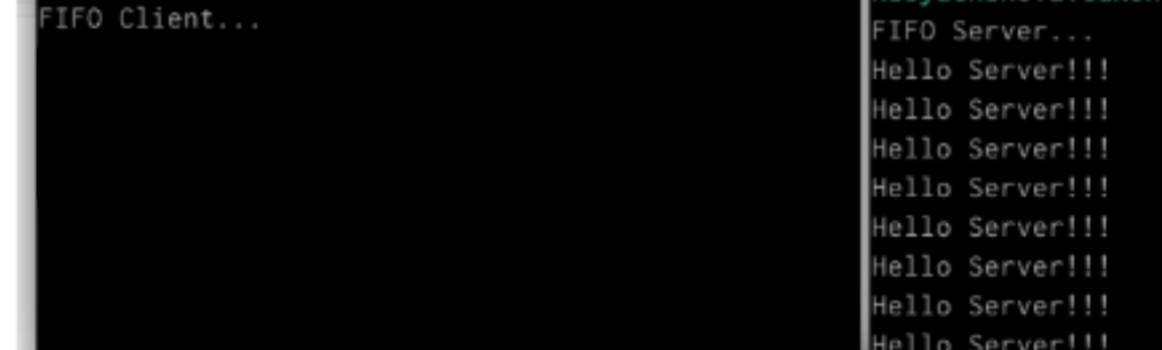
Файл “Makefile”

Далее делаем компиляцию файлов с помощью команды “make all”.(рис. ??).



Компиляция

Затем открываем три терминала для проверки работы наших файлов. В первом пишем “./server”, а в остальных “./client”. В результате каждый терминал вывел по 4 сообщения, а по истечение 30 секунд работа сервера была завершена. Всё работает верно. (рис. ??).



Проверка

# 4 Выводы

В ходе выполнения данной лабораторной работы я приобрела навыки работы с очередями сообщений.

# 5 Ответы на 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. Чтобы создать неименованный канал из командной строкинужно использовать символ |, служащий для объединения двух и более процессов: процесс\_1 |процесс\_2 | процесс\_3…
3. Чтобы создать именованный канал из командной строкинужно использовать либо команду «mknod», либо команду «mkfifo».
4. Неименованный канал является средством взаимодействия между связанными процессами −родительским и дочерним. Родительский процесс создает канал при помощи системного вызова: «int pipe(int fd[2]);». Массив из двух целых чисел является выходным параметром этого системного вызова. Если вызов выполнился нормально, то этот массив содержит два файловых дескриптора. fd[0] является дескриптором для чтения из канала, fd[1] −дескриптором для записи в канал. Когда процесс порождает другой процесс, дескрипторы родительского процесса наследуются дочерним процессом, и, таким образом, прокладывается трубопровод между двумя процессами. Естественно, что один из процессов использует канал только для чтения, а другой −только для записи. Поэтому, если, например, через канал должны передаваться данные из родительского процесса в дочерний, родительский процесс сразу после запуска дочернего процесса закрывает дескриптор канала для чтения, а дочерний процесс закрывает дескриптор для записи. Если нужен двунаправленный обмен данными между процессами, то родительский процесс создает два канала, один из которых используется для передачи данных в одну сторону, а другой −в другую.
5. Файлы именованных каналов создаются функцией mkfifo() или функцией mknod:

* «intmkfifo(constchar\*pathname, mode\_tmode);», где первый параметр − путь, где будет располагаться FIFO (имя файла, идентифицирующего канал), второй параметр определяет режим работы с FIFO (маска прав доступа к файлу),
* «mknod (namefile, IFIFO | 0666, 0)», где namefile −имя канала, 0666 −к каналу разрешен доступ на запись и на чтение любому запросившему процессу),
* «int mknod(const char \*pathname, mode\_t mode, dev\_t dev);».Функцияmkfifo() создает канал и файл соответствующего типа. Если указанный файл канала уже существует, mkfifo() возвращает -1. После создания файла канала процессы, участвующие в обмене данными, должны открыть этот файл либо для записи, любо для чтения.

1. При чтении меньшего числа байтов, чем находится в канале или FIFO, возвращается требуемое число байтов, остаток сохраняется для последующих чтений.При чтении большего числа байтов, чем находится в канале или FIFO, возвращается доступное число байтов. Процесс, читающий из канала, должен соответствующим образом обработать ситуацию, когда прочитано меньше, чем заказано.
2. Запись числа байтов, меньшего емкости канала или FIFO, гарантированно атомарно. Это означает, что в случае, когда несколько процессов одновременнозаписывают 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=ERRPIPE) (если процесс не установил обработки сигнала SIGPIPE, производится обработка по умолчанию −процесс завершается).