Отчёт о выполнении лабораторной работы №15

Российский Университет Дружбы Народов
Факультет Физико-Математических и Естественных Наук

Дисциплина: *Операционные системы*

Работу выполняла: *Арежина Адриана*

№ ст. билета: *1032201674*

Группа: *НКНбд-01-20*

Москва. 2021г.

Цель работы

Приобретение практических навыков работы с именованными каналами.

Задание

Изучите 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?

Выполнение работы

1. Изучила приведённые в тексте программы *server.c* и *client.c*. Создала их у себя. (см. рисунки server, server1, client, client1)

2. Скомпилировала программы, сформировала объектный модуль и получила исполняемые файлы с именами *server* и *client*. (см. рисунки ниже компиляция 1, компиляция 2)

3. Написала аналогичные программы, внеся следующие изменения:

- Работает не 1 клиент, а несколько (например, два). - Клиенты передают текущее время с некоторой периодичностью (например, раз в пять секунд). Использовала функцию *sleep()* для приостановки работы клиента. - Сервер работает не бесконечно, а прекращает работу через некоторое время (например, 30 сек). Использовала функцию *clock()* для определения времени работы сервера. (см. рисунки ниже client2, client2.1, server2, server2.1)

4. Скомпилировала программы, сформировала объектный модуль и получила исполняемые файлы с именами *server2* и *client2*. (см. рисунки ниже компиляция 3, компиляция 4)

5. Если сервер завершит работу, не закрыв канал, файл *FIFO* не удалится, а при следующем запуске файл *FIFO* не создастся, так как он уже существует.

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. Да, командой *pipe*.
3. Да, командой *$ 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. 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. Функция, 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.

Вывод

Я приобрела практические навыки работы с именованными каналами.