Front matter

lang: ru-RU title: Именованные каналы. subtitle: ЛР по ОС №15 author: Танрибергенов Эльдар Марсович group: НПИбд-02-20

Отчёт

о выполнении лабораторной работы № 15.

Именованные каналы.

Выполнил:

студент группы НПИбд-02-20

Танрибергенов Эльдар Марсович.

Студ. билет № 1032208074

Москва 2021 г.

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?

Теоретическое введение

Одним из видов взаимодействия между процессами в операционных системах является обмен сообщениями. Под сообщением понимается последовательность байтов, передаваемая от одного процесса другому. [1]

В операционных системах типа UNIX есть 3 вида межпроцессорных взаимодействий: общеюниксные (именованные каналы, сигналы), System V Interface Definition (SVID — разделяемая память, очередь сообщений, семафоры) и BSD (сокеты).

Для передачи данных между неродственными процессами можно использовать механизм именованных каналов (named pipes). Данные передаются по принципу FIFO (First In First Out) (первым записан — первым прочитан), поэтому они называются также FIFO pipes или просто FIFO. 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.

Каналы представляют собой простое и удобное средство передачи данных, которое, однако, подходит не во всех ситуациях. Например, с помощью каналов довольно трудно организовать обмен асинхронными сообщениями между процессами.

Ход работы:

Полностью скрипты приводить не буду, покажу лишь места, в которые внесены изменения.

Файл server.c

https://github.com/emtanribergenov/OS_labs/blob/master/15/screenshots/server.png

Приём нескольких клиентов обеспечивается через переменную t.

Когда мы получили первое сообщение, мы увеличиваем t на 1 и выходим из цикла read.

И так до тех пор, пока мы не получили t сообщений.

Результат.

https://github.com/emtanribergenov/OS_labs/blob/master/15/screenshots/rser.png

Второй раз клиент был запущен через 6 секунд.

Файл client.c

https://github.com/emtanribergenov/OS_labs/blob/master/15/screenshots/client.png

Получаем текущее время и записываем его в файл.

Потом спим 2 секунды и повторяем операцию.

Результат.

https://github.com/emtanribergenov/OS_labs/blob/master/15/screenshots/rcli.png

Клиент закрывается не сразу

Результат работы скрипта.

https://github.com/emtanribergenov/OS_labs/blob/master/15/screenshots/rs2.png

Скрипт 3

https://github.com/emtanribergenov/OS_labs/blob/master/15/screenshots/s3.png

Результат.

https://github.com/emtanribergenov/OS_labs/blob/master/15/screenshots/rs3.png

Если закрыть сервер нештатно, то временный файл не будет удален. Новый сервер не сможет создать канал, так как он уже будет существовать.

https://github.com/emtanribergenov/OS_labs/blob/master/15/screenshots/last.png#

Вывод:

В результате лабораторной работы я приобрёл практические навыки работы с именованными каналами.

Ответы на вопросы:

1. Именованные каналы, в отличие от неименованных, могут использоваться неродственными процессами. Они дают вам, по сути, те же возможности, что и неименованные каналы, но с некоторыми преимуществами, присущими обычным файлам.
2. Да. Для создания неименованного канала используется системный вызов pipe.
3. Команда /etc/mknod использует системный вызов mknod для создания менованного канала. chmod ug+w NP дает права записи в этот файл любому процессу того же пользователя или любого пользователя из той же группы. line < NP читает одну строку из своего перенаправленного стандартного ввода, т.е. из именованного канала NP, и выводит прочитанное на свой стандартный вывод.
4. #include int pipe(int fd[2J); Здесь содержатся два файловых дескриптора: fd[0] — массив дескрипторов, в котором переменная fd — для чтения, а fd — для записи. Для того чтобы процесс знал, в какой канал писать, а из какого читать информацию, и создаются эти два дескриптора.
5. HANDLE CreateNamedPipe(LPCTSTR lpName, // адрес строки имени канала DWORD dwOpenMode, // режим открытия канала DWORD dwPipeMode, // режим работы канала DWORD nMaxInstances, // максимальное количество //.
6. При чтении большего числа байтов, чем находится в канале или FIFO, 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.
8. Да.
9. Функция write записывает байты счетчика из буфера в файл, связанный с демоном к памяти. Операция записи начинается с текущего положения указателя файла (при наличии), связанного с данным файлом.
10. Быстрый ответ
11. Функция strerror() возвращает строку, описывающую код ошибки, переданный в аргументе errnum, возможно с учетом категории LC_MESSAGES текущей локали для выбора соответсвующего языка.