Міністерство освіти і науки України

Харківський національний університет імені В.Н. Каразіна

Факультет комп’ютерних наук

# ЛАБОРАТОРНА РОБОТА № 12

з дисципліни «Операційні системи»

Тема: «Процеси. Програмні канали (pipe). Іменовані канали (FIFO)»

Виконав:

студент 3 курсу

групи КС-32

Безрук Юрій Русланович

Перевірив: Споров О.Є.

Харків – 2020

Целью данной работы является ознакомление с возможностью организации взаимодействия процессов с помощью неименованных программных каналов (pipe) и именованных программных каналов FIFO.

# ХОД РАБОТЫ

**Задание 1**

В первом задании требуется написать программу, которая организовывает двунаправленную связь между процессом – родителем и процессом – потомком. Родительский процесс считывает строки текста из консоли, и с помощью канала отправляет текст процессу-потомку, который преобразует его к верхнему регистру и с помощью канала отправляет обратно. Процесс-родитель считывает данные, полученные от потомка, выводит их в консоль, и процедура повторяется.

Реализуем задачу таким образом: создаем два массива целых чисел на два элемента, to и from, которые будут представлять собой два канала передачи. To – передача от родителя к потомку, from –от потомка к родителю. Макросами OUT и IN определяем индексы 0 и 1 для «концов» канала.

Создаём два неименованных канала при помощи вызова функции pipe(), передавая ей в аргументы массивы файловых дескрипторов. Далее выполняем функцию fork().

Процесс-потомок работает следующим образом: при запуске он сразу же закрывает для себя выход OUT канала to и выход IN канала from (используя функцию close()), так как с ними будет работать другой процесс и ему они без надобности. Далее он считывает с терминала сообщение функцией fgets и записывает его в канал to при помощи write(). Далее он считывает из канала from ответные данные функцией read() и выводит их, после чего процесс повторяется.

Процесс-потомок реализован похожим образом, только работает с другими выходами каналов, и, соответственно, в другом порядке: сначала считывает, потом записывает. После считывания для полученной строки вызывается функция capitalize(), которая применяет к каждому символу функцию toupper(), переводя его в верхний регистр, если он является буквой.

Остановка работы системы реализована следующим образом. Для остановки необходимо ввести в терминал команду STOP. Процесс-родитель считает её, отправит потомку, и только после отправки проверит. В случае, если введена команда STOP, цикл прерывается, и выходы каналов закрываются. Процесс-потомок же, соответственно, тоже получит такую же команду от родителя и точно так же завершится.

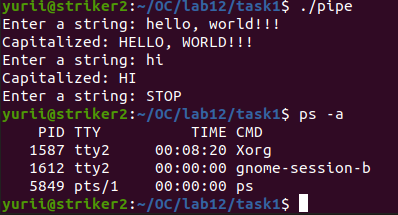


Рисунок 1 – пример работы неименованных каналов

**Задание 2**

Каналы pipe могут работать только с процессами, которые являются родственными по отношению друг к другу. Поэтому для передачи данных между неродственными процессами используются каналы FIFO.

В втором задании нужно реализовать ту же программу для несвязанных между собой процессов используя механизм именованных каналов.

В системе используются две программы: fifo (исходный код в main.c) и second (исходный код в second.c).

Программа-клиент fifo инициализирует две переменных-дескриптора from и to для двух каналов передачи.

Далее процесс обнуляет маску создания файлов текущего процесса вызовом umask(0) для того, чтобы права доступа у создаваемых FIFO точно соответствовали правам, с которыми мы из инициализируем.

Далее при помощи вызова mknod() на диске создается файл для каждого из двух каналов, по пути, который мы передаем. Данный файл будет служить маркером для другого процесса и именно через него станет возможна связь между двумя процессами без родственных «уз».

Каналы открываются функцией open. Канал to – для записи (флаг O\_WRONLY), а канал from – для чтения (O\_RDONLY).

Далее программа заходит в цикл, где делает те же операции, что и в прошлой версии – считывание строки и вывод измененного варианта.

Программа second работает похожим образом, но каналы поменяны местами. Кроме того, программа не создаёт каналы сама, а лишь подключается к уже созданным, поэтому ее следует запускать после первой.

В остальном программа идентична поведению дочернего потока их прошлого задания.

Алгоритм завершения остался неизменен: Невзирая на то, что процессы теперь не связаны, благодаря передаче команды STOP первый процесс может завершить второй. Кроме того, при завершении главной программы удаляются временные файлы каналов вызовом функции unlink().

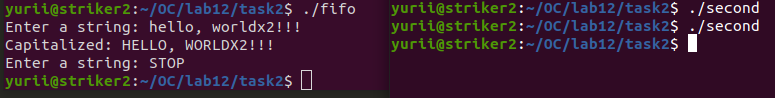


Рисунок 2 - пример работы с именованными каналами

# ВЫВОДЫ

Таким образом, в ходе выполнения данной работы были рассмотрены средства взаимодействия процессов при помощи именованных и неименованных каналов. Исходные, исполняемые и файлы сборки помещены в архив.