## Домашнее задание №2

### Задание:

Организовать серверный сокет на основе механизма FIFO (именованные каналы).

### Особенности задания:

1. Все клиентские приложения знают путь к серверному файлу FIFO.
2. Подключаясь к серверу, клиент передает полный путь к собственному файлу FIFO.
3. Получив от клиента имя файла, сервер создает дочерний процесс, который взаимодействует с клиентом через клиентский FIFO файл, родительский процесс при этом продолжает ожидать подключения новых клиентов.
4. Дочерний процесс сервера начинает передавать клиенту произвольные символы, периодически меняя свой приоритет случайным образом от 0 до 10 (функция **setpriority()**).
5. Клиент должен подсчитать, сколько символов сервер передал ему за 30 секунд.
6. Через 30 секунд клиент должен корректно отключиться.

При сдаче задания так же проверяется возможность одновременной работы нескольких клиентов.

### Теория:

Для создания FIFO используются функции **mknod()** или **mkfifo()**.

Пример:

umask(0);

mknod(FIFO\_FILE, S\_IFIFO | 0777, 0);

Использование umask() необходимо для правильного создания FIFO файла, т.к. по умолчанию в системе установлена маска 022, если создавать файл с правами 0777, то реальные права будут 0755 (т.к. вычитается маска).

Дальнейшая работа с FIFO осуществляется так же как с обычным файлом.

char readbuf[80];

int fp = open(FIFO\_FILE, O\_RDONLY);

read(fp, readbuf, 80);

close(fp);

После окончания работы с каналом желательно его удалить функцией **unlink()**, т.к. FIFO файлы не удаляются автоматически и продолжают существовать в файловой системе.

**Пример случайного изменения приоритета в цикле отправки символов**

if( rand() % 100 > 80 )

setpriority( rand()%11, 0);