#Отчет лабораторной работе № 13

Тема:

Именованные каналы

Российский Университет Дружбы Народов

Факультет Физико-Математических и Естественных Наук

Дисциплина: Операционные системы

Студент: Довлетмурат Байрамгельдыев

Группа: НФИбд-03-20

Москва, 2021г.

Цель работы

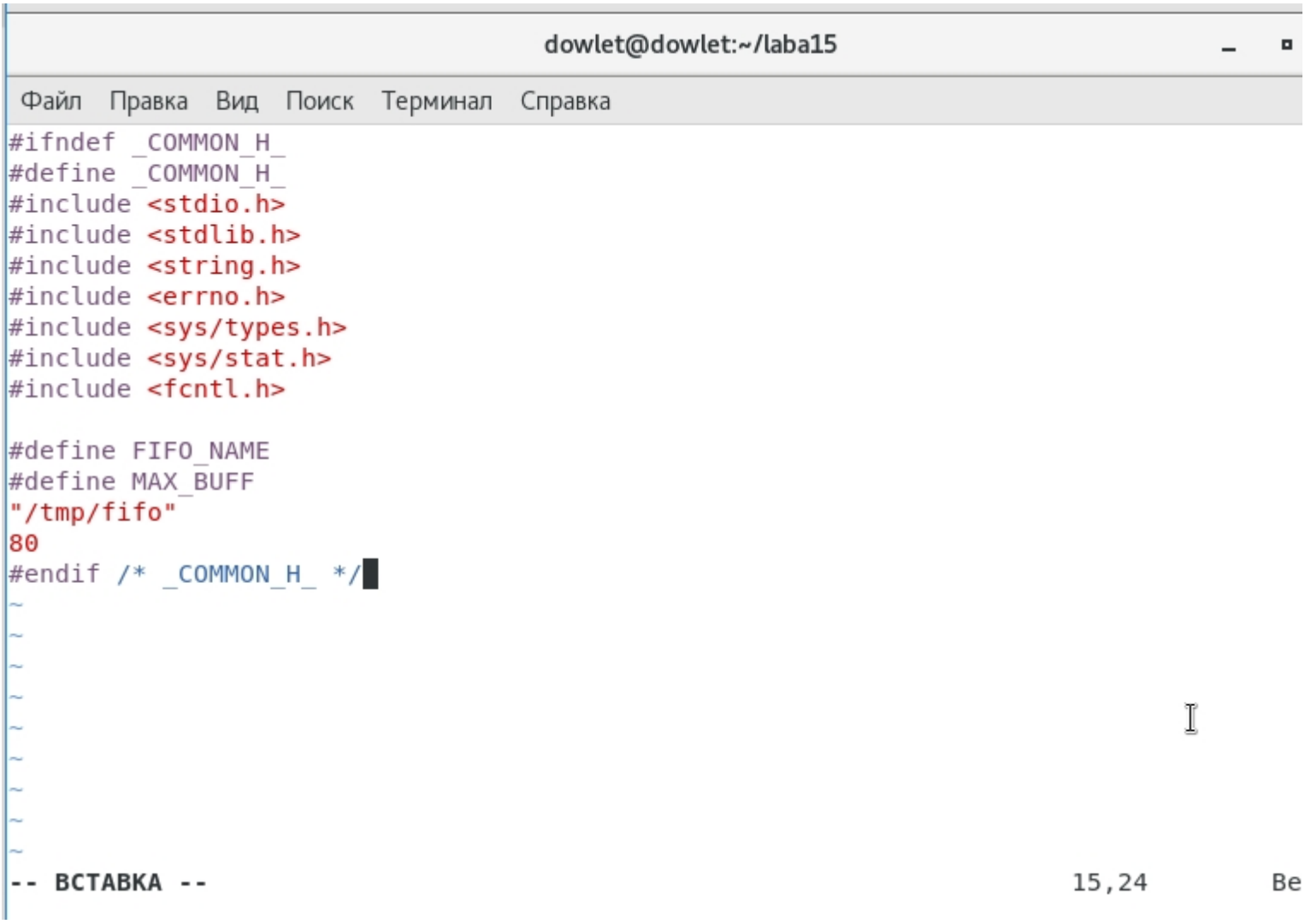
Приобретение практических навыков работы с именованными каналами.

Введение

Именованный канал — один из методов межпроцессного взаимодействия, расширение понятия конвейера в Unix и подобных ОС. Именованный канал позволяет различным процессам обмениваться данными, даже если программы, выполняющиеся в этих процессах, изначально не были написаны для взаимодействия с другими программами. Это понятие также существует и в Microsoft Windows, хотя там его семантика существенно отличается. Традиционный канал — «безымянен», потому что существует анонимно и только во время выполнения процесса. Именованный канал — существует в системе и после завершения процесса. Он должен быть «отсоединён» или удалён, когда уже не используется. Процессы обычно подсоединяются к каналу для осуществления взаимодействия между ними. Одной из самых замечательных возможностей межпроцессорного взаимодействия в Linux являются каналы (pipes) или конвейеры. Чаще всего их применяют для установления связи между процессами.

Ход работы.

1. Изучил приведённые в тексте программу common.h и взял данный пример за образец.



• Изучил приведённые в тексте программу server.c и взял данный пример за образец.

dowlet@dowlet:~/laba15

Файл Правка Вид Поиск Терминал Справка

```
#include "common.h"
int main()
{
int readfd;
int n;
char buff[MAX_BUFF];
printf("FIFO Server...\n");

if(mknod(FIFO_NAME, S_IFIFO | 0666, 0) < 0)
{
fprintf(stderr, "%s: Невозможно создать FIFO (%s)\n",
_FILE_, strerror(errno));
exit(-1);
}

if((readfd = open(FIFO_NAME, O_RDONLY)) < 0)
{
fprintf(stderr, "%s: Невозможно открыть FIFO (%s)\n",
_FILE_, strerror(errno));
exit(-2);
}

while((n = read(readfd, buff, MAX_BUFF)) > 0)
{
-- ВСТАВКА --                                    1,2       Нав
```

• Продолжение кода файла server.c.

dowlet@dowlet:~/laba15

Файл Правка Вид Поиск Терминал Справка

```
{
fprintf(stderr, "%s: Невозможно открыть FIFO (%s)\n",
_FILE_, strerror(errno));
exit(-2);
}

while((n = read(readfd, buff, MAX_BUFF)) > 0)
{

if(write(1, buff, n) != n)
{
fprintf(stderr, "%s: Ошибка вывода (%s)\n",
_FILE_, strerror(errno));
exit(-3);
}
}
close(readfd);

if(unlink(FIFO_NAME) < 0)
{
fprintf(stderr, "%s: Невозможно удалить FIFO (%s)\n",
_FILE_, strerror(errno));
exit(-4);
}
```

-- ВСТАВКА -- 40,2

• Изучил приведённые в тексте программу client.c и взял данный пример за образец.

dowlet@dowlet:~/laba15

Файл Правка Вид Поиск Терминал Справка

```
#include "common.h"
#define MESSAGE "Hello Server!!!\n"
int
main()
{
int writefd;
int msglen;

printf("FIFO Client...\n");

if((writefd = open(FIFO_NAME, O_WRONLY)) < 0)
{
fprintf(stderr, "%s: Невозможно открыть FIFO (%s)\n",
_FILE_, strerror(errno));
exit(-1);
}

msglen = strlen(MESSAGE);
if(write(writefd, MESSAGE, msglen) != msglen)
{
fprintf(stderr, "%s: Ошибка записи в FIFO (%s)\n",
_FILE_, strerror(errno));
exit(-2);
}
```

-- ВСТАВКА -- 1,2 Наве

• Продолжение кода файла client.c.

dowlet@dowlet:~/laba15

Файл Правка Вид Поиск Терминал Справка

```
{
int writefd;
int msglen;

printf("FIFO Client...\n");

if((writefd = open(FIFO_NAME, O_WRONLY)) < 0)
{
fprintf(stderr, "%s: Невозможно открыть FIFO (%s)\n",
_FILE_, strerror(errno));
exit(-1);
}

msglen = strlen(MESSAGE);
if(write(writefd, MESSAGE, msglen) != msglen)
{
fprintf(stderr, "%s: Ошибка записи в FIFO (%s)\n",
_FILE_, strerror(errno));
exit(-2);
}

close(writefd);
exit(0);
}
-- ВСТАВКА --                                                    28,2
```

2. Написал аналогичные программы, внеся следующие изменения: • В коде файла common.h добавил библиотеку time.h, для использования таких функций, как sleep и clock.

dowlet@dowlet:~/laba15

Файл Правка Вид Поиск Терминал Справка

```
#ifndef _COMMON_H_
#define _COMMON_H_
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <time.h>

#define FIFO_NAME "/tmp/fifo"
#define MAX_BUFF 80
#endif /* _COMMON_H_ */
~
~
~
~
~
~
~
~
~
~
```

-- ВСТАВКА -- 11,1

• В код файла server.c так же ввел некоторые изменения

dowlet@dowlet:~/laba15

Файл Правка Вид Поиск Терминал Справка

```
#include "common.h"
int main()
{
int readfd;
int n;
char buff[MAX_BUFF];
printf("FIFO Server...\n");

if(mknod(FIFO_NAME, S_IFIFO | 0666, 0) < 0)
{
fprintf(stderr, "%s: Невозможно создать FIFO (%s)\n",
_FILE_, strerror(errno));
exit(-1);
}
int start = time(NULL);
while (time(NULL) - start <=30){
if((readfd = open(FIFO_NAME, O_RDONLY)) < 0)
{
fprintf(stderr, "%s: Невозможно открыть FIFO (%s)\n",
_FILE_, strerror(errno));
exit(-2);
}

while((n = read(readfd, buff, MAX_BUFF)) > 0)
```

-- ВСТАВКА -- 1,2

• Продолжение кода файла server.c. Сервер работает не бесконечно, а прекращает работу через некоторое время.

dowlet@dowlet:~/laba15

Файл Правка Вид Поиск Терминал Справка

```
exit(-2);
}

while((n = read(readfd, buff, MAX_BUFF)) > 0)
{

if(write(1, buff, n) != n)
{
fprintf(stderr, "%s: Ошибка вывода (%s)\n",
_FILE_, strerror(errno));
exit(-3);
}
}
close(readfd);

if(unlink(FIFO_NAME) < 0)
{
fprintf(stderr, "%s: Невозможно удалить FIFO (%s)\n",
_FILE_, strerror(errno));
exit(-4);
}
exit(0);
}
}
-- ВСТАВКА --                                                    44,2
```

• В коде файла client.c использовал функцию sleep() для приостановки работы клиента. Так же приравнял переменную ttime к нулю, задав его в нутри цикла.

dowlet@dowlet:~/laba15

Файл Правка Вид Поиск Терминал Справка

```
#include "common.h"
#define MESSAGE "Hello Server!!!\n"
int
main()
{
int writefd;
int msglen;

printf("FIFO Client...\n");

if((writefd = open(FIFO_NAME, O_WRONLY)) < 0)
{
fprintf(stderr, "%s: Невозможно открыть FIFO (%s)\n",
_FILE_, strerror(errno));
exit(-1);
}
int i;
for (i=0; i<10; i++){
sleep(10);
long ttime = time(NULL);
msglen = strlen(ctime(&ttime));
if(write(writefd, ctime(&ttime), msglen) != msglen)
{
fprintf(stderr, "%s: Ошибка записи в FIFO (%s)\n",
```

-- ВСТАВКА -- 1,2 Н

• Продолжение кода файла client.c.

dowlet@dowlet:~/laba15

Файл Правка Вид Поиск Терминал Справка

```
printf("FIFO Client...\n");

if((writefd = open(FIFO_NAME, O_WRONLY)) < 0)
{
fprintf(stderr, "%s: Невозможно открыть FIFO (%s)\n",
_FILE_, strerror(errno));
exit(-1);
}
int i;
for (i=0; i<10; i++){
sleep(10);
long ttime = time(NULL);
msglen = strlen(ctime(&ttime));
if(write(writefd, ctime(&ttime), msglen) != msglen)
{
fprintf(stderr, "%s: Ошибка записи в FIFO (%s)\n",
_FILE_, strerror(errno));
exit(-2);
}

close(writefd);
exit(0);
}
```

-- ВСТАВКА -- 3

• Запустил makefile.

dowlet@dowlet:~/laba15

Файл Правка Вид Поиск Терминал Справка

```
all: server client

server: server.c common.h
	gcc server.c -o server

client: client.c common.h
	gcc client.c -o client

clean:
	-rm server client *.o
```

• Проверяю, запустил коды в файлах server.c и client.c

```
[dowlet@dowlet laba15]$ ./server

FIFO Server...
server.c: Невозможно создать FIFO (File exists)
```

• В случае, если сервер завершит работу, не закрыв канал, файл FIFO не удалится, поэтому его в следующий раз создать будет нельзя и вылезет ошибка, следовательно, работать ничего не будет.

```
[dowlet@dowlet laba15]$ ./client

FIFO Client...
```

Вывод:

Приобрел практические навыки работы с именованными каналами.

Библиография

https://ru.wikipedia.org/wiki/%D0%98%D0%BC%D0%B5%D0%BD%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D1%8B%D0%B9_%D0%BA%D0%B0%D0%BD%D0%B0%D0%BB

http://www.rusmedvedev.ru/articles/view?id=20

https://habr.com/ru/post/122108/

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. Создание неименованного канала из командной строки невозможно.
3. Создание именованного канала из командной строки возможно.
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) ; mkfifo(FIFO_NAME, 0600) ; 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).
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 много направленная 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. Write - 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. Строковая функция strerror - функция языков C/C++, транслирующая код ошибки, который обычно хранится в глобальной переменной errno, в сообщение об ошибке, понятном человеку. Ошибки эти возникают при вызове функций стандартных Сибиблиотек. 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.