РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет Физико-Математических Наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЁТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ № 15

дисциплина: Операционные системы

Студент: Манаева Варвара Евгеньевна

Группа: НФИбд-01-20

Преподаватель: Кулябов Дмитрий Сергеевич

МОСКВА

2021 г.

Техническое оснащение:

- Персональный компьютер с операционной системой Windows 7;
- Планшет для записи видеосопровождения и голосовых комментариев;
- Виртуальная коробка VirtualBox, виртуальная машина с установленной на ней операционной системой CentOS;
- Microsoft Teams, использующийся для записи скринкаста лабораторной работы;
- Приложение MarkPad 2 для редактирования файлов формата *md*;
- *pandoc* для конвертации файлов отчётов и презентаций.

Объект и предмет исследования: Именованные каналы.

Цель [1]: Приобретение практических навыков работы с именованными каналами.

Задачи:

1) Изучить теоретическую справку из текста лабораторной [1];
2) Написать программы клиента и сервера из справки [1];
3) Дополнить их по заданию.

Теоретические вводные данные [1]:

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

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.

Файлы именованных каналов создаются функцией mkfifo(3).

```
#include <sys/types.h>
#include <sys/stat.h>
int mkfifo(const char *pathname, mode_t mode);
```

Первый параметр — имя файла, идентифицирующего канал, второй параметр — маска прав доступа к файлу.

После создания файла канала процессы, участвующие в обмене данными, должны открыть этот файл либо для записи, либо для чтения. При закрытии файла сам канал продолжает существовать. Для того чтобы закрыть сам канал, нужно удалить его файл, например с помощью вызова unlink(2).

Рассмотрим работу именованного канала на примере системы клиент–сервер.
Сервер создаёт канал, читает из него текст, посылаемый клиентом, и выводит его на терминал.
Вызов функции mkfifo() создаёт файл канала (с именем, заданным макросом FIFO_NAME):

```
mkfifo(FIFO_NAME, 0600);
```

В качестве маски доступа используется восьмеричное значение 0600, разрешающее процессу с аналогичными реквизитами пользователя чтение и запись. Можно также установить права доступа 0666.
Открываем созданный файл для чтения:

```
f = fopen(FIFO_NAME, O_RDONLY);
```

Ждём сообщение от клиента. Сообщение читаем с помощью функции read() и печатаем на экран. После этого удаляется файл FIFO_NAME и сервер прекращает работу.

Клиент открывает FIFO для записи как обычный файл:

```
f = fopen(FIFO_NAME, O_WRONLY);
```

Посылаем сообщение серверу с помощью функции write().
Для создания файла FIFO можно использовать более общую функцию mknod(2), предназначенную для создания специальных файлов различных типов (FIFO, сокеты, файлы устройств и обычные файлы для хранения данных).

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
int mknod(const char *pathname, mode_t mode, dev_t dev);
```

Тогда, вместо

```
mkfifo(FIFO_NAME, 0600);
```

пишем

```
mknod(FIFO_NAME, S_IFIFO | 0600, 0);
```

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.

3. Пример программы

3.1. Файл common.h

```
/*
* common.h - заголовочный файл со стандартными определениями
*/
#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>
#define FIFO_NAME "/tmp/fifo"
#define MAX_BUFF 80
#endif /* __COMMON_H__ */
```

3.2. Файл server.c

```
/*
* server.c - реализация сервера
*
* чтобы запустить пример, необходимо:
* 1. запустить программу server на одной консоли;
* 2. запустить программу client на другой консоли.
*/
#include "common.h"
int
main()
{
int readfd; /* дескриптор для чтения из FIFO */
int n;
char buff[MAX_BUFF]; /* буфер для чтения данных из FIFO */
/* баннер */
printf("FIFO Server...\n");
/* создаем файл FIFO с открытыми для всех
```

```
* правами доступа на чтение и запись
*/
if(mknod(FIFO_NAME, S_IFIFO | 0666, 0) < 0)
{
fprintf(stderr, "%s: Невозможно создать FIFO (%s)\n",
__FILE__, strerror(errno));
exit(-1);
}
/* откроем FIFO на чтение */
if((readfd = open(FIFO_NAME, O_RDONLY)) < 0)
{
fprintf(stderr, "%s: Невозможно открыть FIFO (%s)\n",
__FILE__, strerror(errno));
exit(-2);
}
/* читаем данные из FIFO и выводим на экран */
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); /* закроем FIFO */
/* удалим FIFO из системы */
if(unlink(FIFO_NAME) < 0)
{
fprintf(stderr, "%s: Невозможно удалить FIFO (%s)\n",
__FILE__, strerror(errno));
exit(-4);
}
exit(0);
}
```

3.3. Файл client.c

```
/*
* client.c - реализация клиента
*
* чтобы запустить пример, необходимо:
* 1. запустить программу server на одной консоли;
* 2. запустить программу client на другой консоли.
*/
#include "common.h"
#define MESSAGE "Hello Server!!!\n"
int
main()
{
int writefd; /* дескриптор для записи в FIFO */
int msglen;
/* баннер */
printf("FIFO Client...\n");
/* получим доступ к FIFO */
if((writefd = open(FIFO_NAME, O_WRONLY)) < 0)
{
fprintf(stderr, "%s: Невозможно открыть FIFO (%s)\n",
__FILE__, strerror(errno));
exit(-1);
}
/* передадим сообщение серверу */
msglen = strlen(MESSAGE);
if(write(writefd, MESSAGE, msglen) != msglen)
{
fprintf(stderr, "%s: Ошибка записи в FIFO (%s)\n",
```

```
__FILE__, strerror(errno));
exit(-2);
}
/* закроем доступ к FIFO */
close(writefd);
exit(0);
}
```

3.4. Файл Makefile

```
all: server client
server: server.c common.h
gcc server.c -o server88
client: client.c common.h
gcc client.c -o client
clean:
-rm server client *.o
```

Этапы работы [1]:

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. В случае, если сервер завершит работу, не закрыв канал, будет ошибка.

!

Выводы: По окончании данной лабораторной работы я приобрела практических навыков работы с именованными каналами.

Контрольные вопросы [1]:

1) В чем ключевое отличие именованных каналов от неименованных?

Ответ: Именованные каналы отличаются от не именованных наличием идентификатора канала, который представлен как специальный файл.

2) Возможно ли создание неименованного канала из командной строки?

Ответ: нет

3) Возможно ли создание именованного канала из командной строки?

Ответ: да, например с помощью функции mkfifo(FIFO_NAME,MODE) в терминале.

4) Опишите функцию языка C, создающую неименованный канал.

Ответ: int pipe(int fd[2]) - 2 файловых дескриптора (чтение и запись).

5) Опишите функцию языка C, создающую именованный канал.

Ответ: mkfifo(FIFO_NAME,MODE)

6) Что будет в случае прочтения из fifo меньшего числа байтов, чем находится в канале? Большего числа байтов?

Ответ: Смотря на пример из лаб15 -> произойдет ошибка при чтении.

7) Аналогично, что будет в случае записи в fifo меньшего числа байтов, чем позволяет буфер? Большего числа байтов?

Ответ: Смотря на пример из лаб15 -> произойдет ошибка при записи.

8) Могут ли два и более процессов читать или записывать в канал?

Ответ: При технологии FIFO да, но это будет неудобно. Более удобный вариант - один на чтение, один на запись.

9) Опишите функцию write (тип возвращаемого значения, аргументы и логику работы). Что означает 1 (единица) в вызове этой функции в программе server.c (строка 42)?

Ответ: write имеет следущую логику:

```
write(fd, buffer, count), где buffer - записываемые файлы; count - байты;  fd - file descriptor.
```

Например,

```
write(1,buff,n) - 1-fd запись(0-чтение, 1-запись); buff- записываемые данные; n - кол-во записываемых данных.
```

10) Опишите функцию strerror.

Ответ: strerror() возвращает указатель на сообщение об ошибке, связанное с номером ошибки (errno- number of error).

Библиография:

1. Текст лабораторной работы № 15.