Отчёт по лабораторной работе №14

Именованные каналы

Киньябаева Аиша Иделевна

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Целью данной работы является приобретение практических навыков работы с именованными каналами

2 Задание

Создание командных файлов

3 Выполнение лабораторной работы

Пишу первую программу с работой нескольких клиентов(рис. 3.1)

```c
#include "common.h"
#include <time.h>

#define MESSAGE_LEN 64

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);
  }

  while(1) {
    char buff[MESSAGE_LEN];
    time_t current_time = time(NULL);
    snprintf(buff, MESSAGE_LEN, "Client at %ld: Hello Server!!!\n", current_time);
    /* передадим сообщение серверу */
    msglen = strlen(buff);
    if (write(writefd, buff, msglen) != msglen) {
      fprintf(stderr, "%s: Ошибка записи в FIFO (%s)\n", __FILE__,
              strerror(errno));
      exit(-2);
    }
    sleep(5);
  }

  /* закроем доступ к FIFO */
  close(writefd);
  exit(0);
}
```

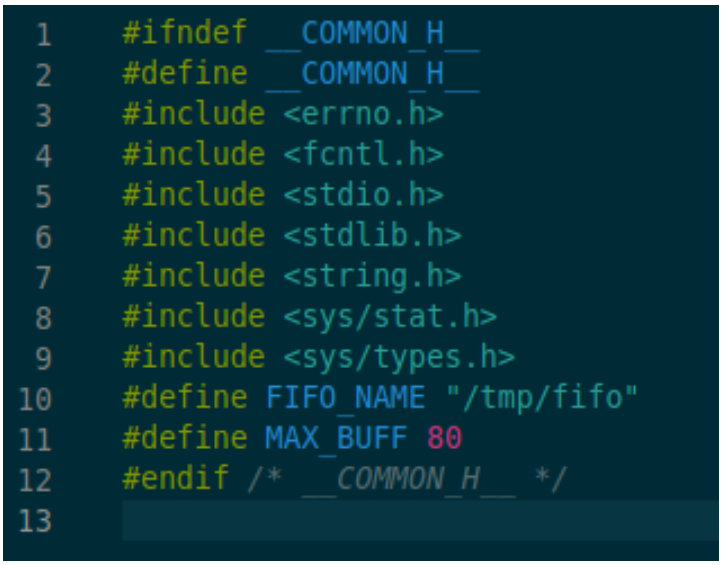
Рис. 3.1: 1 задание

Пишу вторую программу-сервер (рис. 3.2)

```c
#include "common.h"
#include <time.h>

#define MAX_CLIENTS 2
#define MESSAGE_LEN 64
#define MAX_BUFF 1024

int main() {
  int readfds[MAX_CLIENTS]; /* массив дескрипторов для чтения из FIFO */
  int n;
  char buff[MAX_BUFF]; /* буфер для чтения данных из FIFO */
  fd_set fdset;
  int active_clients = 0;
  clock_t start = clock();

  /* 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);
  }

  /* открываем FIFO на чтение */
  for (int i = 0; i < MAX_CLIENTS; i++) {
    if ((readfds[i] = open(FIFO_NAME, O_RDONLY)) < 0) {
      fprintf(stderr, "%s: Невозможно открыть FIFO %d (%s)\n", __FILE__, i,
          strerror(errno));
      exit(-2);
    }
    active_clients ++;
  }
```

Рис. 3.2: 2 задание

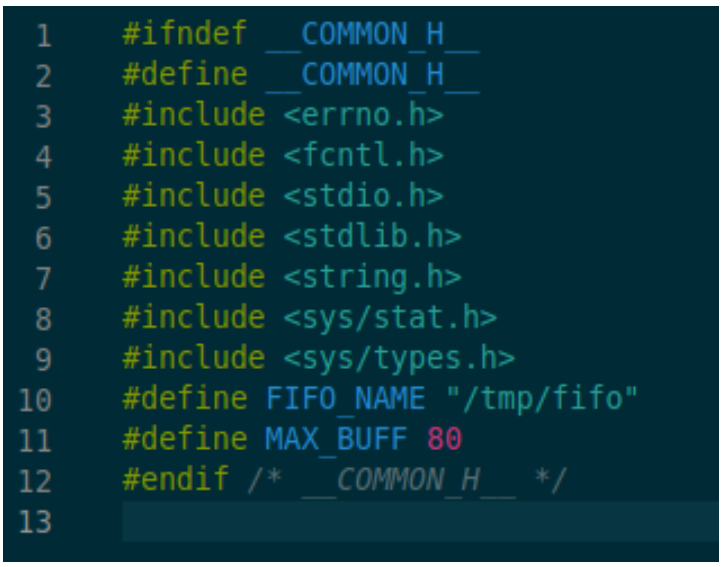
Исполняемый файл (рис. 3.3)



```c
#ifndef __COMMON_H__
#define __COMMON_H__
#include <errno.h>
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/stat.h>
#include <sys/types.h>
#define FIFO_NAME "/tmp/fifo"
#define MAX_BUFF 80
#endif /* __COMMON_H__ */
```

Рис. 3.3: Результат

Мейкфайл(рис. 3.4)



```
#ifndef __COMMON_H__
#define __COMMON_H__
#include <errno.h>
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/stat.h>
#include <sys/types.h>
#define FIFO_NAME "/tmp/fifo"
#define MAX_BUFF 80
#endif /* __COMMON_H__ */
```

Рис. 3.4: 3 задание

Ну и в завершение работы выкладываю все на гит

4 Контрольные вопросы

1. Ошибка в этой строке заключается в использовании квадратных скобок. В командной оболочке Bash они используются для выполнения условных выражений вместе с командой test. Правильно будет так: while (($1 != “exit”))

2. В bash для объединения нескольких строк в одну можно использовать оператор конкатенации + или .. Например: “ string1=“Hello” string2=“world” result=$string1string2 echo $result “

Вывод будет: Helloworld

3. Утилита seq в Linux используется для создания итераторов, которые генерируют последовательности чисел. Она имеет следующий синтаксис: seq ОПЦИИ… ПОСЛЕДНЕЕ Например, команда “seq 1 5” выведет последовательность чисел от 1 до 5. Некоторые из опций, которые можно использовать с утилитой seq: -f –format=ФОРМАТ используйте указанный формат для каждого числа. -s –separator=СЕПАРАТОР используйте указанный разделитель между числами

4. Результат вычисления выражения будет равен 3, так как при использовании символов двойных круглых скобок происходит округление до ближайшего целого числа.

5. Некоторые отличия командной оболочки zsh от bash включают:

• Мощная автодополнение и расширенная подсказка, которые делают работу с командами более эффективной и быстрой. • Zsh имеет более интуитивный синтаксис для управления массивами и строками, что делает ее удобным выбором для обработки текста. • В zsh доступен “глубокий патчинг”, позволяющий изменять внутреннюю логику самой оболочки. • Zsh имеет более развитую систему конфигурации и управления плагинами, что делает ее наиболее гибкой и настраиваемой оболочкой в мире Linux.

6. Синтаксис верный, но надо уточнить значение переменной LIMIT. Также, важно помнить, что в данном цикле нет тела цикла - то есть ничего не будет происходить, пока не будет добавлен код, который будет выполняться внутри цикла.

7. C++:

ПРЕИМУЩЕСТВА BASH: - Встроенная поддержка системных команд и утилит - Незаменим при работе с системными скриптами на Unix-подобных операционных системах - Гибкость и легкость в написании коротких скриптов, относительно C++

НЕДОСТАТКИ BASH: - Недостаточная мощность в сравнении с C++ - Ограниченный функционал - Сложные программы сложнее считывать и понимать в bash, чем в C++

5 Выводы

В ходе данной лабораторной работы были изучены основны программирования и созданы командные файлы.