Цельработы

Приобретение практичсекихнавыков работы с именованными каналами.

Ход работы

Изучили приведенные в тексте программы server.c client.c **.Взяв данные примеры за образец, написали аналогичные программы,внесяслдующ ие изменения**

> **Работаетдва клиента,а не один.Клиенты передаюттекущ ее времяс некоторой периодичностью .Использовали функцию sleep() для приостановки работы клиента.Сервер работает не бесконечно,а прекращ ает работу через 30 секунд.Использовали функцию clock() дляопределениявремени работы сервера.**

Вывод

Приобрели практические навыки работы с именованными каналами.

Контрольныевопросы

.Именованные каналы отличаютсяотнеименованныхналичием идентификатора канала, который представлен какспециальный файл (соответственно имяименованного канала — это имяфайла).

. Для создания неименованного канала используется системный вызов pipe.Массив из двухцелыхчисел являетсявыходным параметром этого системного вызова.

.Вы можете создаватьименованные каналы изкомандной строки и внутри программы.С давних времен программой создания их в командной строке была команда:m knod - $ m knod имя_файла ,однако команды m knod нет в списке команд X/Open,поэтому она включена не во все UNIX-подобные системы.Предпочтительнее применять в командной строке - $ m kfifo имя_файла.

. intread(intpipe_fd,void *area,intcnt);

Intwrite(intpipe_fd,void *area,intcnt);

Первый аргументэтихвызовов - дескриптор канала,второй - указательна областьпамяти,с которой происходитобмен,третий - количество байт.Оба вызова возвращ аютчисло переданных байт(или -1 - при ош ибке).

. intm kfifo (constchar*pathnam e,m ode_tm ode);Первый параметр — имя файла, идентифицирующ его канал,второй параметр маска прав доступа кфайлу.Вызов функции mkfifo() создаёт файл канала (с именем,заданным макросом FIFO_NAM E):m kfifo(FIFO_NAM E, 0600);

.При чтении меньш его числа байтов,чем находитсяв канале,возвращ аетсятребуемое число байтов,остатоксохраняетсядляпоследующ ихчтений.При чтении больш его числа байтов, чем находитсяв канале или FIFO возвращ аетсядоступное число байтов.

. При записи больш его числа байтов,чем это позволяет канал или FIFO,вызов write(2) блокируетсядо освобожденияитребуемого места.При этом атомарностьоперации не гарантируется.Если процесс пытаетсязаписатьданные в канал,не открытый ни одним процессом на чтение,процессу генерируетсясигнал.Записьчисла байтов,меньш его емкости канала или FIFO,гарантированно атомарно.Это означает,что в случае,когда несколько процессов одновременно записываютв канал,порции данныхотэтихпроцессов не перемеш иваются.

.Вобщ ем случае возможна многонаправленнаяработа процессов с каналом,т.е.возможна ситуация,когда с одним и тем же каналом взаимодействуютдва и более процесса,и каждый извзаимодействующ ихканалов пиш ети читаетинформацию в канал.Но традиционной

схемой организации работы с каналом являетсяоднонаправленнаяорганизация,когда канал связываетдва,в больш инстве случаев,или несколько взаимодействующ ихпроцесса,каждый из которыхможетлибо читать,либо писатьв канал.

. W rite - Функция записывает length байтов из буфера bufferв файл,определенный дескриптором файла fd.Эта операция чисто 'двоичная'и без буферизации.Реализуется как непосредственный вызов DOS.С помощ ью функции write мы посылаем сообщ ение клиенту или серверу.

. Строковая функция strerror- функция языков C/C++,транслирующ ая код ош ибки,который обычно хранится в глобальной переменной errno,в сообщ ение об ош ибке,понятном человеку. Ош ибки эти возникаютпри вызове функций стандартныхСи-библиотек.

Возвращ енный указательссылаетсяна статическую строку с ош ибкой,котораяне должна быть изменена программой.Дальнейш ие вызовы функции strerrorперезапиш ут содержание этой строки. Интерпретированные сообщ енияоб ош ибкахмогутразличаться,это зависитотплатформы и компилятора.

Приложение

Рис 2

```c
#include "common.h"

#define MESSAGE "client1!!\n"

int main(){
    for(int i = 0; i < 10; ++i){
        FILE* writefd;
        int msglen;
        long int ttime;
        ttime = time(NULL);

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

        if((writefd = fopen(FIFO_NAME, "w+")) == NULL){
            fprintf(stderr, "%s; It's impossible to open FIFO (%s)\n", __FILE__, strerror(errno));
            exit(-1);
        }

        msglen = strlen(MESSAGE);
```

Рис.3

```c
#include "common.h"

#define MESSAGE "client2!!!\n"

int main(){
    for(int i = 0; i < 10; ++i){
        FILE* writefd;
        int msglen;
        long int ttime;
        ttime = time(NULL);
                (char [16])"FIFO CLIENT...\n"
        printf("FIFO CLIENT...\n");

        if((writefd = fopen(FIFO_NAME, "w+")) == NULL){
            fprintf(stderr, "%s; It's impossible to open FIFO (%s)\n", __FILE__, strerror(errno));
            exit(-1);
        }

        msglen = strlen(MESSAGE);

        if(fwrite(MESSAGE, msglen, 1, writefd) != 1){
            fprintf(stderr, "%s: FIFO write error (%s)\n", __FILE__, strerror(errno));
            exit(-2);
        }

        sleep(4);
        fclose(writefd);
    }
    exit(0);
}
```

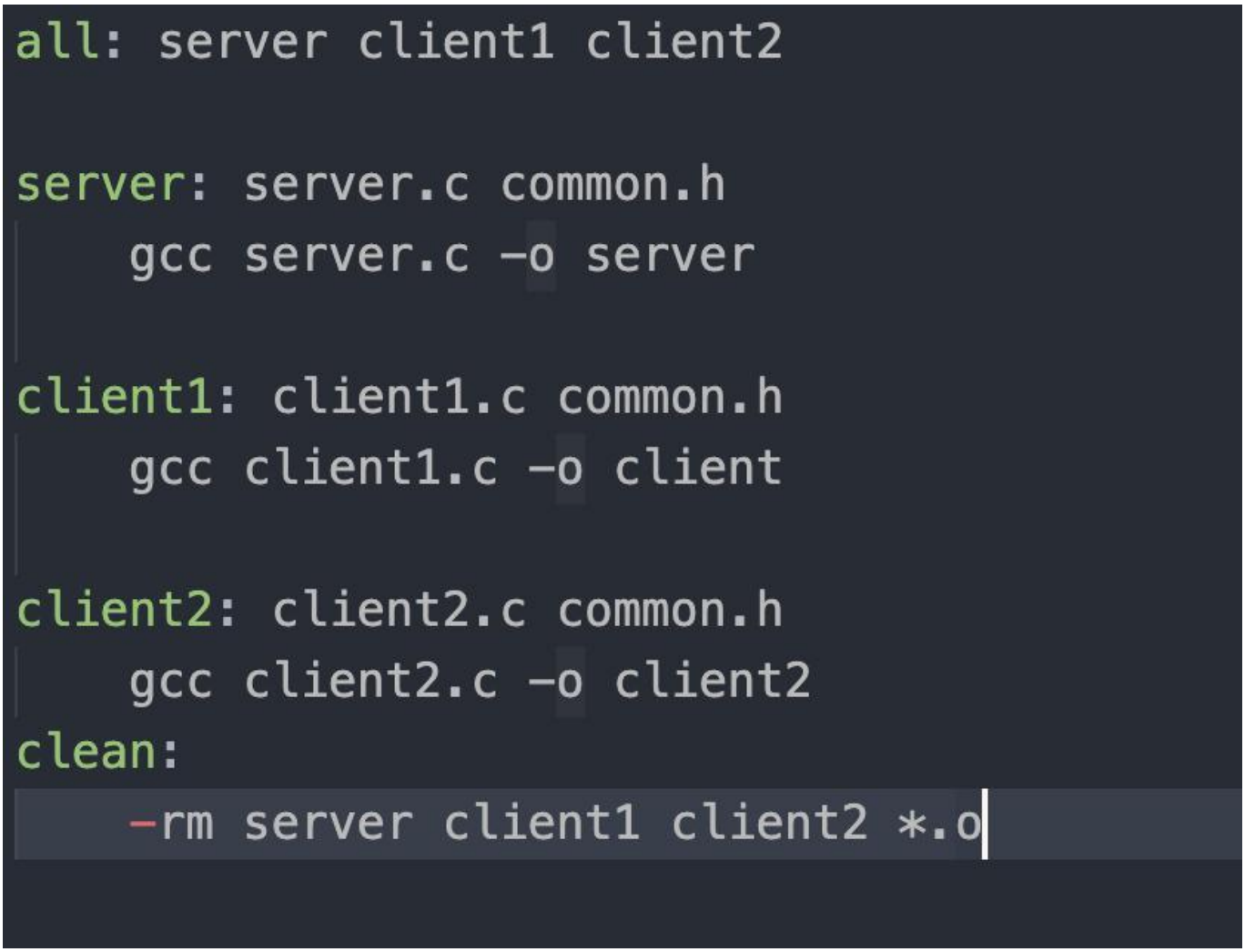
Рис.4

```c
#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>
#include <zconf.h>

#define FIFO_NAME "/Users/dimashestakov/fifo"
#define MAX_BUFF 1024


#endif /* __COMMON_H__ */
```

Рис.5

```c
#include "common.h"

int main() {
    FILE* readfd = NULL;
    int n;
    char buff[MAX_BUFF];

    printf("FIFO SERVER...\n");
    if(mknod(FIFO_NAME, S_IFIFO | 0600,0)<0){
        fprintf(stderr, "%s: It's impossible to create FIFO (%s)\n", __FILE__, strerror(errno));
        exit(-1);
    }

    if((readfd = fopen(FIFO_NAME, "r")) == NULL){
        fprintf(stderr, "%s: It's impossible to open FIFO (%s)\n", __FILE__, strerror(errno));
        exit(-2);
    }

    clock_t now = time(NULL), start = time(NULL);

    while(now-start < 30){
        if((n = fread(buff, MAX_BUFF, 1, readfd)) > 0){
            if(fwrite(buff, MAX_BUFF, n, readfd) != n)
            {
                fprintf(stderr, "%s: Input error (%s)\n", __FILE__, strerror(errno));
                exit(-3);
            }
        }
        now = time(NULL);
    }

    printf("\nServer timeout\n%lu seconds passed", (now - start));
    fclose(readfd);

    if(remove(FIFO_NAME) != 0)
    {
        fprintf(stderr, "%s: It's impossible to delete FIFO (%s)\n", __FILE__, strerror(errno));
        exit(-4);
    }
    exit(0);
}
```

Рис.6



```
all: server client1 client2

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

client1: client1.c common.h
    gcc client1.c -o client

client2: client2.c common.h
    gcc client2.c -o client2
clean:
    -rm server client1 client2 *.o
```

Рис.7