Лабораторная работа №15

Покрас Илья Михайлович

Цель работы

Приобретение практических навыков работы с именованными каналами.

Последовательность работы

1. Я изучил приведённые в тексте программы server.c и client.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/socket.h>
#include <sys/un.h>
#define SOCKET_NAME "/tmp/echo.server"
#define MAX_BUFF 80
#endif
```

Common.h

```c
#include "common.h"
#define MESSAGE "Hello Server!!!"
int main (){
  char buf[MAX_BUFF];
  struct sockaddr_un serv_addr;
  struct sockaddr_un clnt_addr;
  int sockfd;
  int saddrlen;
  int caddrlen;
  int msglen;
  int n;
  bzero(&serv_addr, sizeof(serv_addr));
  serv_addr.sun_family=AF_UNIX;
  strcpy(serv_addr.sun_path,SOCKET_NAME);
  saddrlen=sizeof(serv_addr.sun_family)+strlen(serv_addr.sun_path);

  bzero(&clnt_addr, sizeof(clnt_addr));
  clnt_addr.sun_family=AF_UNIX;
  strcpy(clnt_addr.sun_path,"/tmp/clnt.XXXXXX");
  mktemp(clnt_addr.sun_path);
  caddrlen=sizeof(clnt_addr.sun_family)+strlen(clnt_addr.sun_path);
```

Server.c

```c
saddrlen=sizeof(serv_addr.sun_family)+strlen(serv_addr.sun_path);
if(bind(sockfd,(struct sockaddr*)&serv_addr,saddrlen)<0)
  {
    fprintf(stderr,"%s:Error associating socket with address(%s)\n",
            __FILE__,strerror(errno));
    exit(-2);
  }
max_caddrlen=sizeof(clnt_addr);
for( ; ; )
  {
    int n;
    int caddrlen;
    caddrlen=max_caddrlen;
    n=recvfrom(sockfd,buf,MAX_BUFF,0,(struct sockaddr *)&clnt_addr,&caddrlen);
    if(n<0)
      {
        fprintf(stderr,"%s:Reception error(%s)\n",
                __FILE__,strerror(errno));
        exit(-3);
      }
```

Server.c(2)

```
      printf("echo:%s\n",buf);
      msglen=strlen(MESSAGE);
      if(sendto(sockfd,MESSAGE,msglen,0,(struct sockaddr *)&clnt_addr,caddrlen)!=n)
        {
          fprintf(stderr,"%s:Transmission error(%s)\n",
                  __FILE__,strerror(errno));
          exit(-5);
        }
    }
  close(sockfd);
  unlink(clnt_addr.sun_path);
  exit(0);
}
```

Server.c(3)

```
CC=gcc
CFLAGS=
programs=server client
all: $(programs)

server: server.c common.h
        $(CC) $(CFLAGS) $< -o $@

client: client.c common.h
        $(CC) $(CFLAGS) $< -o $@

clean:
        -rm $(programs) *.o
```

Makefile

2. Взяв примеры в тексте лабораторной работы за образец, написал аналогичные программы, внеся следующие изменения: Работает не 1 клиент, а несколько (например, два). Клиенты передают текущее время с некоторой периодичностью (например, раз в пять секунд). Используйте функцию sleep() для приостановки работы клиента. 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, то при следующем запуске сервера он работать не будет.

```c
#include "common.h"
#define MESSAGE "Hi Server!!!"
int main (){
  char buf[MAX_BUFF];
  struct sockaddr_un serv_addr;
  struct sockaddr_un clnt_addr;
  int sockfd;
  int saddrlen;
  int caddrlen;
  int msglen;
  int n;
  bzero(&serv_addr, sizeof(serv_addr));
  serv_addr.sun_family=AF_UNIX;
  strcpy(serv_addr.sun_path,SOCKET_NAME);
  saddrlen=sizeof(serv_addr.sun_family)+strlen(serv_addr.sun_path);

  bzero(&clnt_addr, sizeof(clnt_addr));
  clnt_addr.sun_family=AF_UNIX;
  strcpy(clnt_addr.sun_path,"/tmp/clnt.XXXXXX");
  mktemp(clnt_addr.sun_path);
  caddrlen=sizeof(clnt_addr.sun_family)+strlen(clnt_addr.sun_path);
```

Client 2

```c
#include "common.h"
#define MESSAGE "Good day Server!!!"
int main (){
  char buf[MAX_BUFF];
  struct sockaddr_un serv_addr;
  struct sockaddr_un clnt_addr;
  int sockfd;
  int saddrlen;
  int caddrlen;
  int msglen;
  int n;
  bzero(&serv_addr, sizeof(serv_addr));
  serv_addr.sun_family=AF_UNIX;
  strcpy(serv_addr.sun_path,SOCKET_NAME);
  saddrlen=sizeof(serv_addr.sun_family)+strlen(serv_addr.sun_path);

  bzero(&clnt_addr, sizeof(clnt_addr));
  clnt_addr.sun_family=AF_UNIX;
  strcpy(clnt_addr.sun_path,"/tmp/clnt.XXXXXX");
  mktemp(clnt_addr.sun_path);
  caddrlen=sizeof(clnt_addr.sun_family)+strlen(clnt_addr.sun_path);
```

Client 3

```
Unix Sockets Server...
echo:Hello Server!!!
echo:Hi Server!!!!!!
server.c:Transmission error(No such file or directory)
```

Server status

```
echo:Hello Client!!!
```

Client 2 status

Client 3 status

Вывод

Я приобретел практические навыки работы с именованными каналами.

Контрольные вопросы:

1. BSD является сокращением от ‘Berkeley Software Distribution’, названия, которое было выбрано Berkeley CSRG (Computer Systems Research Group) для их дистрибутива Unix.

2. Сокет (socket) - это конечная точка сетевых коммуникаций. Он является чем-то вроде “портала”, через которое можно отправлять байты во внешний мир. Приложение просто пишет данные в сокет. Программирование сокетов в Linux , их дальнейшая буферизация, отправка и транспортировка осуществляется используемым стеком протоколов и сетевой аппаратурой. Чтение данных из сокета происходит аналогичным образом. В программе сокет идентифицируется дескриптором - это просто переменная типа int. Программа получает дескриптор от операционной системы при создании сокета, а затем передаёт его сервисам socket API для указания сокета, над которым необходимо выполнить то или иное действие

3.Именованные каналы, описанные в главе 11, очень похожи на сокеты, но в способах их использования имеются значительные различия.

· Именованные каналы могут быть ориентированными на работу с сообщениями, что значительно упрощает программы.

· Именованные каналы требуют использования функций ReadFile и WriteFile, в то время как сокеты могут обращаться также к функциям send и recv.

· В отличие от именованных каналов сокеты настолько гибки, что предоставляют пользователям возможность выбрать протокол для использования с сокетом, например, TCP или UDP. Кроме того, пользователь имеет возможность выбирать протокол на основании характера предоставляемой услуги или иных факторов.

· Сокеты основаны на промышленном стандарте, что обеспечивает их совместимость с системами, отличными от Windows.

Имеются также различия в моделях программирования сервера и клиента.

4. Коммуникационный домен определяет форматы адресов и правила их интерпретации. Внутри них существуют сокеты.

5. Виды сокетов:

· Сокеты в файловом пространстве имён (file namespace, сокеты Unix) используют в

качестве адресов имена файлов специального типа.

· Сокеты в файловом пространстве имён похожи на именованные каналы тем, что для идентификации сокетов используются файлы специального типа. В мире сокетов есть и аналог неименованных каналов — парные сокеты.

· Сетевой сокет – сокет, в котором формат адреса имеет вид ip(7). Поскольку адрес транспортного уровня состоит из пары ip-адрес: порт, то и в структуре

под адрес отводится два поля.

6. Когда поддержка BSD сокетов были добавлена в ядро Linux, разработчики решили добавить их единовременно все 17 (на сегодня 20) сокетных вызовов, и добавили для этих вызовов один дополнительный уровень косвенности. Для всей группы этих вызовов введен один новый, редко упоминаемый, системный вызов:

int socketcall(int call, unsigned long *args),

где:

— call — численный номер сетевого вызова (SYS_CONNECT, SYS_ACCEPT…);

— args — указатель 6-ти элементного массива (блок параметров), в который последовательно упакованы все параметры любого из системных вызовов этой группы (сетевой), без различения их типа (приведенные к unsigned long)

7. Базовая эталонная модель взаимодействия открытых систем (БЭМВОС) – это концептуальная основа, определяющая характеристики и средства открытых систем. Она обеспечивает работу в одной сети систем, выпускаемых различными производителями. Разработана ISO (международной организацией стандартов) и широко используется во всём мире как основа концепций информационных сетей и их ассоциаций. На базе этой модели описываются правила и процедуры передачи данных между открытыми системами. Она также описывает структуру открытой системы и комплекс стандартов, которым она должна удовлетворять. Основными элементами модели являются: уровни, объекты, соединения, физические средства соединений.

Модель информационной системы состоит из трёх основных составляющих:

- прикладные процессы (осуществляют обработку данных);
- область взаимодействия (размещаемые в ней блоки прокладывают в сети логические каналы (пунктирная линия на рисунке) между портами прикладных процессов и обеспечивает их взаимодействие);
- физические средства соединений (обеспечивают физическую связь систем).