学号:031602523 姓名:刘宏岩　学院:数计学院　专业:计算机类 班级:实验班

《Linux 操作系统设计实践》

实验三:网络编程

实验环境:Ubuntu16.04

实验内容:

1. 套接字 socket：

网络上的两个程序通过一个双向的通信连接实现数据的交换，这个连接的一端称为一个 socket。Socket 的英文原义是“孔”或“插座”。作为 BSD UNIX 的进程通信机制，取后一种意思。通常也称作"套接字"，用于描述 **IP 地址和端口**，是一个通信链的句柄，可以用来实现不同虚拟机或不同计算机之间的通信。在 Internet 上的主机一般运行了多个服务软件，同时提供几种服务。每种服务都打开一个 Socket，并绑定到一个端口上，不同的端口对应于不同的服务。Socket 正如其英文原意那样，像一个多孔插座。一台主机犹如布满各种插座的房间，每个插座有一个编号， 客户软件将插头插到不同编号的插座，就可以得到不同的服务。

2. 相关 API 学习

socket 的接口函数声明在头文件 sys/types.h 和 sys/socket.h 中

1） socket——创建套接字

该函数用来创建一个套接字，并返回一个描述符，该描述符可以用来访问该套接字，它的原型如下：

```
int socket(int domain, int type, int protocol);
```

函数中的三个参数分别对应前面所说的三个套接字属性。protocol 参数设置为 0 表示使用默认协议

2） bind——命名（绑定）套接字

该函数把通过 socket 调用创建的套接字命名，从而让它可以被其他进程使用。对于 AF_UNIX，调用该函数后套接字就会关联到一个文件系统路径名，对于 AF_INET，则会关联到一个 IP 端口号。函数原型如下：

```
int bind( int socket, const struct sockaddr *address, size_t address_len);
```

成功时返回 0，失败时返回-1；

3） Sendto——发送数据

该函数把缓冲区 buffer 中的信息给送给指定的 IP 端口的程序，原型如下：

```
int sendto(int sockfd, void *buffer, size_t len, int flags, struct sockaddr *to, socklen_t tolen);
```

buffer 中储存着将要发送的数据，len 是 buffer 的长度，而 flags 在应用中通常被设置为 0，to 是要发送数据到的程序的 IP 端口，tolen 是 to 参数的长度。成功时返回发送的数据的字节数，失败时返回-1.

4） Recvfrom——接收数据

该函数把发送给程序的信息储存在缓冲区 buffer 中，并记录数据来源的程序 IP 端口，原型如下：

```
int recvfrom(int sockfd, void *buffer, size_t len,  int flags, struct sockaddr *src_from, socklen_t *src_len);
```

buffer 用于储存接收到的数据，len 指定 buffer 的长度，而 flags 在应用中通常被设置 0，src_from 若不为空，则记录数据来源程序的 IP 端口，若 src_len 不为空，则其长度信息记录在 src_len 所指向的变量中。

注意：默认情况下，recvfrom 是一个阻塞的调用，即直到它接收到数据才会返回。

5） Close——关闭 socket

该系统调用用来终止服务器和客户上的套接字连接，我们应该总是在连接的两端（服务器和客户）关闭套接字。

3. 服务器端

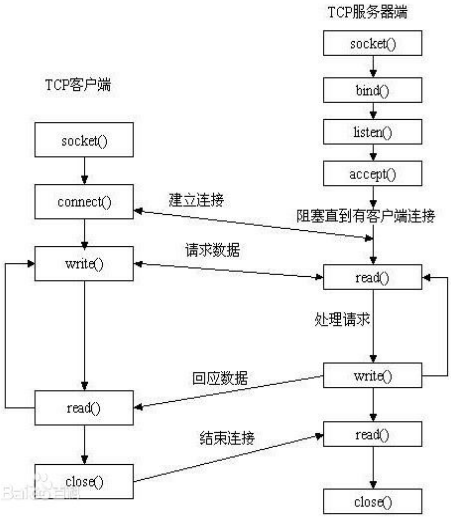
- 服务器应用程序用系统调用 socket 来创建一个套接口，它是系统分配给该服务器进程的类似文件描述符的资源，它**不能与其他的进程共享。**
- 接下来，服务器进程会给套接字起个名字（监听），我们使用系统调用 bind 来给套接字命名。然后服务器进程就开始等待客户连接到这个套接字。
- 然后系统调用 recvfrom 来接收来自客户程序发送过来的数据。服务器程序对数据进行相应的处理，再通过系统调用 sendto 把处理后的数据发送回客户程序。

4. 客户端

- 基于数据报 socket 的客户端比服务器端简单，同样，客户应用程序首先调用 socket 来创建一个未命名的套接字，与服务器一样，客户也是通过 sendto 和 recvfrom 来向服务器发送数据和从服务器程序接收数据。

实验总结:

1. 实验流程图



2. 服务器端代码

```
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
int main()
{
        int server_sockfd = -1;
        int server_len = 0;
        int client_len = 0;
        char buffer[512];
        int result = 0;
        struct sockaddr_in server_addr;
        struct sockaddr_in client_addr;
        //创建数据报套接字
        server_sockfd = socket(AF_INET, SOCK_DGRAM, 0);
        //设置监听 IP 端口
        server_addr.sin_family = AF_INET;
        server_addr.sin_addr.s_addr = htonl(INADDR_ANY);
        server_addr.sin_port = htons(9739);
        server_len = sizeof(server_addr);
        //绑定（命名）套接字
        bind(server_sockfd, (struct sockaddr*)&server_addr, server_len);
        //忽略子进程停止或退出信号
        signal(SIGCHLD, SIG_IGN);
        while(1)
        {
                //接收数据，用 client_addr 来储存数据来源程序的 IP 端口
                result = recvfrom(server_sockfd, buffer, sizeof(buffer), 0,
                                (struct sockaddr*)&client_addr, &client_len);
                if(fork() == 0)
                {
                        //利用子进程来处理数据
                        buffer[0] += 'a' - 'A';
                        sleep(5);
                        //发送处理后的数据
                        sendto(server_sockfd, buffer, sizeof(buffer),0 ,
                                (struct sockaddr*)&client_addr, client_len);
                        printf("%c\n", buffer[0]);
                        //注意，一定要关闭子进程，否则程序运行会不正常
                        exit(0);
                }
        }
        //关闭套接字
        close(server_sockfd);
}
```

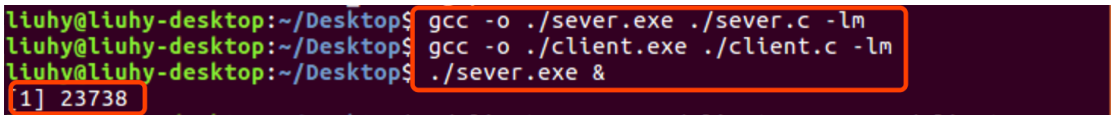
3. 客户端代码

```
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <stdlib.h>
#include <stdio.h>

int main(int agrc, char *argv[])
{
        struct sockaddr_in server_addr;
        int server_len = 0;
        int sockfd = -1;
        int result = 0;
        char c = 'A';
        //取第一个参数的第一个字符
        if(agrc > 1)
                c = argv[1][0];
        //创建数据报套接字
        sockfd = socket(AF_INET, SOCK_DGRAM, 0);
        //设置服务器 IP 端口
        server_addr.sin_family = AF_INET;
        server_addr.sin_addr.s_addr = inet_addr("127.0.0.1");
        server_addr.sin_port = htons(9739);
        server_len = sizeof(server_addr);
        //向服务器发送数据
        sendto(sockfd, &c, sizeof(char), 0,
                (struct sockaddr*)&server_addr, server_len);
        //接收服务器处理后发送过来的数据，由于不关心数据来源，所以把后两个参数设为 0
        recvfrom(sockfd, &c, sizeof(char), 0, 0, 0);
        printf("char from server = %c\n", c);
        //关闭套接字
        close(sockfd);
        exit(0);
}
```

4. 实验截图

1. 运行服务器



2. 客户端发送数据，大写转化为小写