

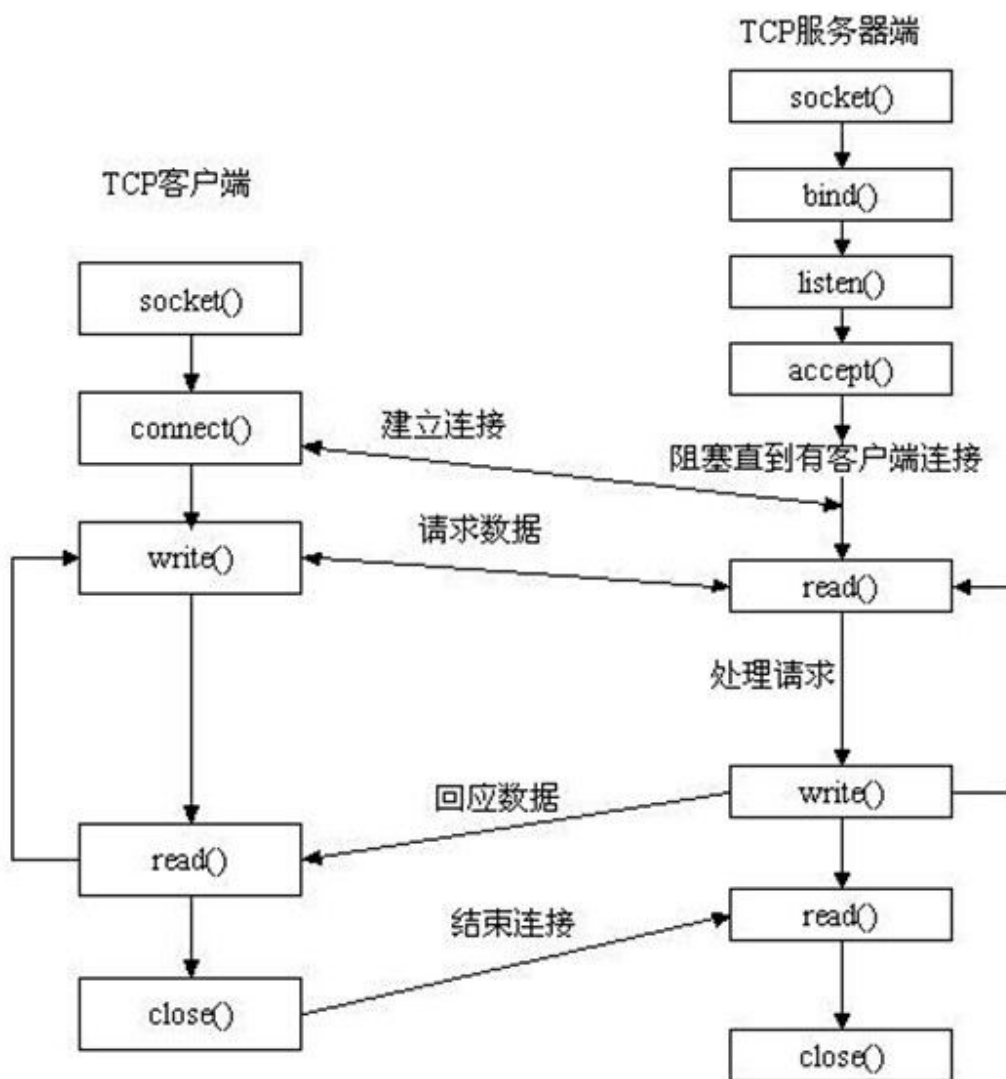
# iTOP-4412 实现基于 TCP 的 socket 编程

大家好，今天我们来学习一下 linux 网络通信程序的编写，我们使用的硬件平台是 iTOP-4412 开发板。

TCP 是一种面向连接的、可靠的、基于 IP 的传输层协议。通过 TCP 可以保证我们传送的数据的正确性。

Linux 下网络通信程序基本上都是采用 socket 的方式。socket 起源于 Unix，而 Unix/Linux 基本哲学之一就是“一切皆文件”，都可以用“打开 open->读写 read/write->关闭 close”模式来操作。Socket 就是该模式的一个实现，socket 即是一种特殊的文件，一些 socket 函数就是对其进行的操作（读/写 IO、打开、关闭）。说白了 socket 是应用程序与 TCP/IP 协议族通信的中间软件抽象层，它是一组接口。

现在我们看一下基于 TCP/IP 应用程序通信的流程，如下图：



通过上图我们可以看到 TCP/IP 通信是基于服务器/客户端的模式来实现的，首先是服务器（server）端调用 `socket` 函数创建一个套接字，然后调用 `bind` 绑定函数，绑定函数主要是设置通信时使用哪种地址族（IPv4，IPv6 等），使用的端口号。然后调用 `listen` 函数来监听客户端的连接请求。

现在我们来看下客户端（client）端的流程，首先调用 `socket` 函数创建一个套接字，然后调用 `connect` 函数连接服务器，这时服务器端的 `listen` 函数监听到客户端的连接请求就会调用 `accept` 函数去接受请求，这样连接就建立好了。之后双方就可以调用 `read/write` 函数收发数据了，在完成通信以后服务器（server）和客户端（client）调用 `close` 函数关闭创建的套接字。

下面我们来看一个实现 TCP/IP 的通信的例子，首先来看一下服务器（server）端的代码：

```
#include <stdlib.h>

#include <sys/types.h>

#include <stdio.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <string.h>


int main()
{
    int sfp, nfp, num = 0;

    struct sockaddr_in s_add,c_add;

    int sin_size;

    unsigned short portnum=0x8888;


    char buffer[100] = {0};


    printf("Hello,welcome to my server !\r\n");

    /* 创建 TCP 连接的套接字 */

    sfp = socket(AF_INET, SOCK_STREAM, 0);

    if(-1 == sfp)
```

```
{
```

```
    printf("socket fail ! \r\n");
```

```
    return -1;
```

```
}
```

```
    printf("socket ok !\r\n");
```

```
    /* 变量 s_add 清零 */
```

```
    bzero(&s_add,sizeof(struct sockaddr_in));
```

```
    s_add.sin_family=AF_INET;
```

```
    s_add.sin_addr.s_addr=htonl(INADDR_ANY);
```

```
    s_add.sin_port=htons(portnum);
```

```
    /* 绑定 s_add 到套接字 sfp 上 */
```

```
    if(-1 == bind(sfp,(struct sockaddr *)&s_add, sizeof(struct sockaddr)))
```

```
{
```

```
    printf("bind fail !\r\n");
```

```
    return -1;
```

```
}
```

```
    printf("bind ok !\r\n");
```

```
/*监听函数，静听客户端的连接请求 */
```

```
if(-1 == listen(sfp,5))
```

```
{
```

```
    printf("listen fail !\r\n");
```

```
    return -1;
```

```
}
```

```
printf("listen ok\r\n");
```

```
sin_size = sizeof(struct sockaddr_in);
```

```
/* 接受连接请求 */
```

```
nfp = accept(sfp, (struct sockaddr *)&c_add, &sin_size);
```

```
if(-1 == nfp)
```

```
{
```

```
    printf("accept fail !\r\n");
```

```
    return -1;
```

```
}
```

```
printf("accept ok!\r\nServer start get connect from %#x : %#x\r\n",
      ntohl(c_add.sin_addr.s_addr), ntohs(c_add.sin_port));

while(1)
{
    memset(buffer, 0, 100);

    sprintf(buffer, "hello,welcome to my server(%d) \r\n", num++);

    /* 发送函数 */

    send(nfp, buffer, strlen(buffer), 0);

    usleep(500000);
}

/* 关闭 socket 连接 */

close(nfp);

/* 关闭 socket 连接 */

close(sfp);

return 0;
}
```

程序首先是包含一些需要用到的头文件 ,然后是 main 主函数 ,在 main 函数里面首先是定义了一些变量 ,

然后调用 `socket` 函数创建一个套接字，`socket` 函数的第二个参数是 `SOCK_STREAM`，表示创建的是 TCP 连接。然后调用 `bzero` 函数把变量 `s_add` 清零，然后给 `s_add` 结构里面的变量赋值：

```
s_add.sin_family=AF_INET;//使用 IPv4 协议
```

```
s_add.sin_addr.s_addr=htonl(INADDR_ANY);//允许任何地址
```

```
s_add.sin_port=htons(portnum);//设置端口号
```

然后调用 `bind` 绑定函数，使用的是 IPv4 协议族，然后调用 `listen` 监听函数，监听用户的连接请求。在监听到用户的请求后调用 `accept` 函数接受请求，然后进入到循环发送的代码，我们会循环发送

“hello,welcome to my server” + 发送次数号，最后会调用 `close` 关闭套接字。

下面我们来看看客户端（client）端的代码：

```
#include <stdlib.h>
```

```
#include <stdio.h>
```

```
#include <sys/socket.h>
```

```
#include <netinet/in.h>
```

```
#include <string.h>
```

```
int main(int argc, char **argv)
```

```
{
```

```
    int cfd;
```

```
    int recbyte;
```

```
    int sin_size;
```

```
    char buffer[1024] = {0};
```

```
struct sockaddr_in s_add, c_add;
```

```
unsigned short portnum = 0x8888;
```

```
printf("Hello,welcome to client!\r\n");
```

```
if(argc != 2)
```

```
{
```

```
    printf("usage: echo ip\n");
```

```
    return -1;
```

```
}
```

```
/* 创建一个 TCP 连接的 socket */
```

```
cfd = socket(AF_INET, SOCK_STREAM, 0);
```

```
if(-1 == cfd)
```

```
{
```

```
    printf("socket fail ! \r\n");
```

```
    return -1;
```

```
}
```

```
printf("socket ok !\r\n");
```



```
/* 变量 s_add 清零 */
```

```
bzero(&s_add,sizeof(struct sockaddr_in));
```

```
s_add.sin_family=AF_INET;
```

```
s_add.sin_addr.s_addr= inet_addr(argv[1]);
```

```
s_add.sin_port=htons(portnum);
```

```
printf("s_addr = %#x ,port : %#x\r\n",s_add.sin_addr.s_addr,s_add.sin_port);
```

```
/* 连接服务器函数 */
```

```
if(-1 == connect(cfd,(struct sockaddr *)&s_add, sizeof(struct sockaddr)))
```

```
{
```

```
printf("connect fail !\r\n");
```

```
return -1;
```

```
}
```

```
printf("connect ok !\r\n");
```

```
while(1)
```

```
{
```

```
/* 接收服务器发过来的数据 */
```

```
if(-1 == (recbyte = read(cfd, buffer, 1024)))
```

```
{
```

```
printf("read data fail !\r\n");
```

```
return -1;
```

```
}
```

```
printf("read ok\r\nREC:\r\n");
```

```
buffer[recbyte]='\0';
```

```
printf("%s\r\n",buffer);
```

```
}
```

```
/* 关闭套接字 */
```

```
close(cfd);
```

```
return 0;
```

```
}
```

首先是包含一些需要的头文件，然后进入 main 主函数定义了一些变量，然后调用 socket 函数创建套接字，然后调用 bzero 函数把变量 s\_add 清零，然后给 s\_add 结构里面的变量赋值：

```
s_add.sin_family=AF_INET;//使用 IPv4 协议
```

```
s_add.sin_addr.s_addr= inet_addr(argv[1]);//设置要连接的 IP 地址(这里是我们执行程序的时候传递进来的)
```

```
s_add.sin_port=htons(portnum);//设置端口号
```

然后调用 connect 函数来连接服务器 ( server )，在连接成功后，就进入了循环接收函数，使用 read 函数接收服务器发送的数据。最后会调用 close 函数关闭套接字。

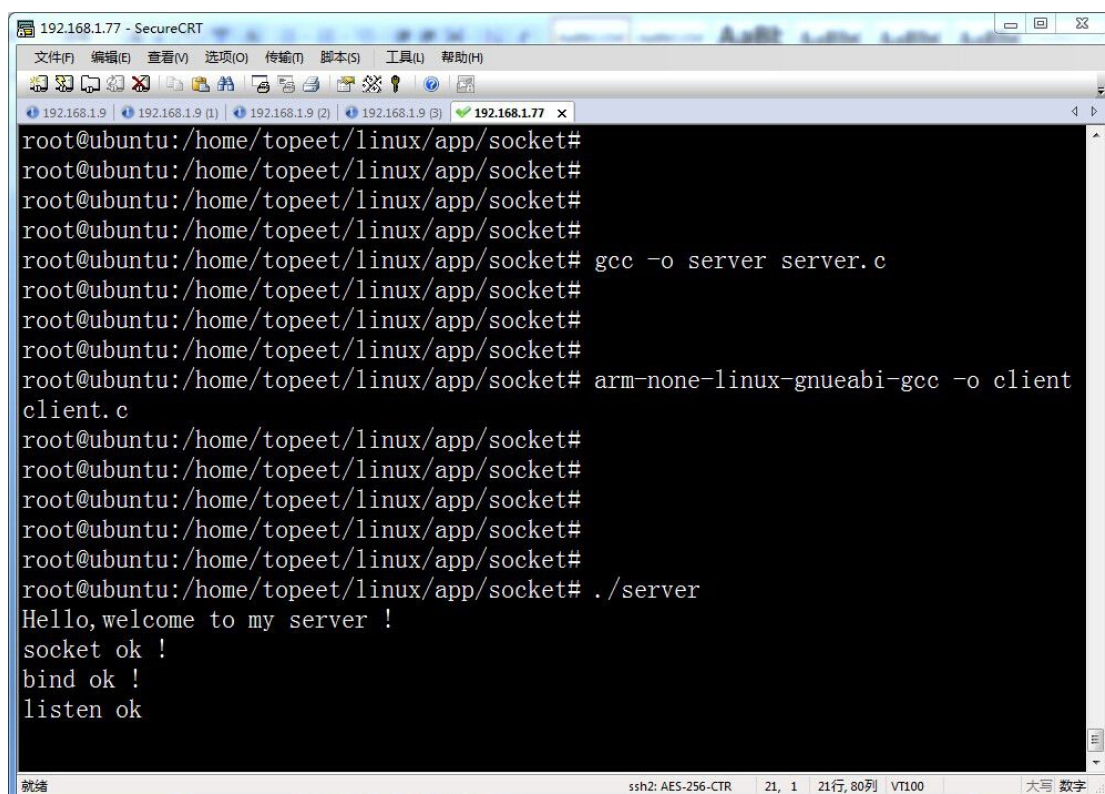
下面我们来编译下这两个程序，服务器 ( server ) 的程序我们运行在虚拟机 Ubuntu 上，所以使用下面的命令编译：

```
gcc -o server server.c
```

这样就生成了 server 可执行文件，客户端 ( client ) 的程序我们运行在 iTOP-4412 开发板上，我们使用下面的命令编译：

```
arm-none-linux-gnueabi-gcc -o client client.c
```

这样就生成了 client 可执行程序，把 client 下载到 iTOP-4412 开发板上，现在我们开始运行这两个程序，首先在虚拟机 Ubuntu 上运行 server 程序，如下图：



```
192.168.1.77 - SecureCRT
文件(F) 编辑(E) 查看(V) 选项(O) 传输(T) 脚本(S) 工具(I) 帮助(H)
192.168.1.9 | 192.168.1.9 (1) | 192.168.1.9 (2) | 192.168.1.9 (3) | 192.168.1.77 x
root@ubuntu:/home/topeet/linux/app/socket#
root@ubuntu:/home/topeet/linux/app/socket#
root@ubuntu:/home/topeet/linux/app/socket#
root@ubuntu:/home/topeet/linux/app/socket# gcc -o server server.c
root@ubuntu:/home/topeet/linux/app/socket#
root@ubuntu:/home/topeet/linux/app/socket#
root@ubuntu:/home/topeet/linux/app/socket# arm-none-linux-gnueabi-gcc -o client
client.c
root@ubuntu:/home/topeet/linux/app/socket#
root@ubuntu:/home/topeet/linux/app/socket#
root@ubuntu:/home/topeet/linux/app/socket#
root@ubuntu:/home/topeet/linux/app/socket#
root@ubuntu:/home/topeet/linux/app/socket# ./server
Hello,welcome to my server !
socket ok !
bind ok !
listen ok
```

我们可以看到 server 打印出来的运行信息，现在 server 运行到了 listen 函数开始监听客户端的连接。下面我

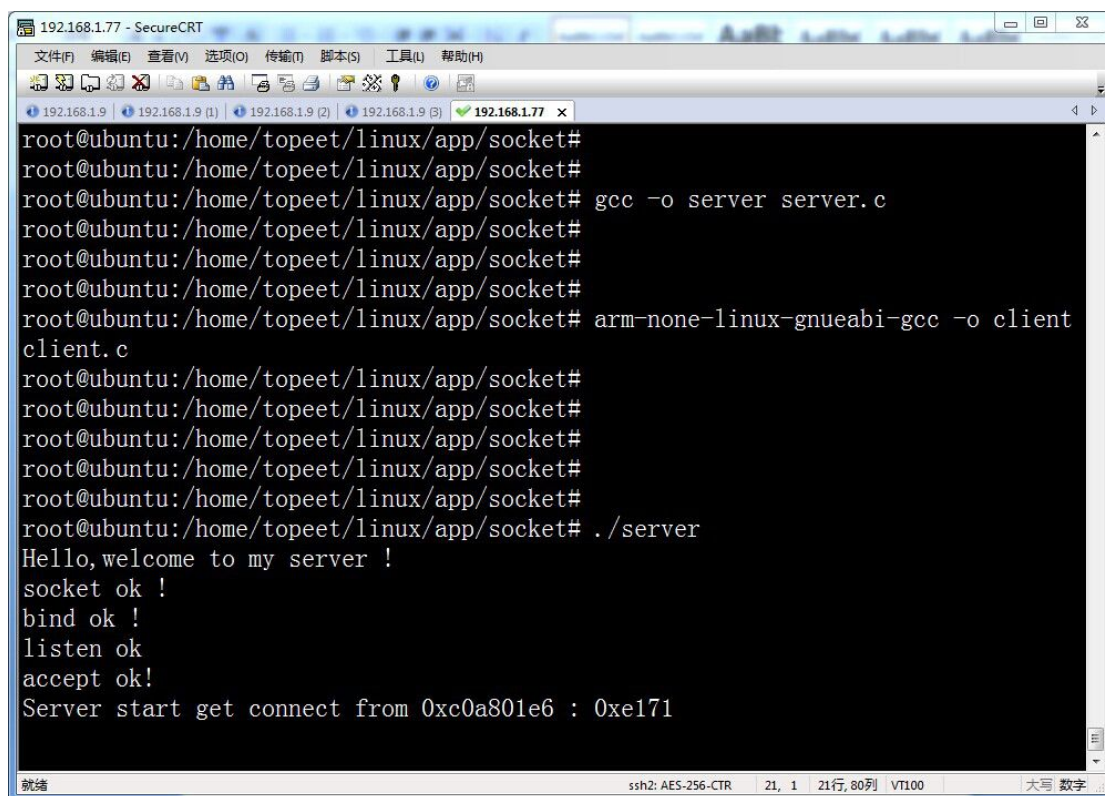
们在 iTOP-4412 开发板上运行 client 程序（因为我把 client 下载到了/bin 目录下，所以先进入到/bin 目录）

执行下面的命令：

```
./client 192.168.1.77
```

上面命令里面的 192.168.1.77 是我们虚拟机 Ubuntu 的 IP 地址，我们看到程序连接成功，首先看一下虚拟机

Ubuntu 上的 server 打出的信息，如下图：



```
192.168.1.77 - SecureCRT
文件(F) 编辑(E) 查看(V) 选项(O) 传输(T) 脚本(S) 工具(I) 帮助(H)
192.168.1.9 | 192.168.1.9 (1) | 192.168.1.9 (2) | 192.168.1.9 (3) | 192.168.1.77 x
root@ubuntu:/home/topeet/linux/app/socket#
root@ubuntu:/home/topeet/linux/app/socket#
root@ubuntu:/home/topeet/linux/app/socket# gcc -o server server.c
root@ubuntu:/home/topeet/linux/app/socket#
root@ubuntu:/home/topeet/linux/app/socket#
root@ubuntu:/home/topeet/linux/app/socket# arm-none-linux-gnueabi-gcc -o client
client.c
root@ubuntu:/home/topeet/linux/app/socket#
root@ubuntu:/home/topeet/linux/app/socket#
root@ubuntu:/home/topeet/linux/app/socket#
root@ubuntu:/home/topeet/linux/app/socket#
root@ubuntu:/home/topeet/linux/app/socket# ./server
Hello, welcome to my server !
socket ok !
bind ok !
listen ok
accept ok!
Server start get connect from 0xc0a801e6 : 0xe171
就绪 ssh2: AES-256-CTR 21, 1 21行, 80列 VT100 大写 数字
```

我们可以看到上图中 server 打印出了客户端的 ip 地址和端口号 “Server start get connect from 0xc0a801e6 : 0xe171”。

然后我们看一下 iTOP-4412 开发板串口的打印信息，如下图：

```
[root@iTOP-4412]#  
[root@iTOP-4412]# cd /bin/  
[root@iTOP-4412]#  
[root@iTOP-4412]#  
[root@iTOP-4412]# ./client 192.168.1.77  
Hello,welcome to client!  
socket ok !  
s_addr = 0x4d01a8c0 ,port : 0x8888  
connect ok !  
read ok  
REC:  
hello,welcome to my server(0)  
  
read ok  
REC:  
hello,welcome to my server(1)  
  
read ok  
REC:  
hello,welcome to my server(2)  
  
read ok  
REC:  
hello,welcome to my server(3)
```

通过上图我们可以看到打印连接成功 “connect ok !”，然后串口会一直打印

read ok

REC:

hello,welcome to my server(0)

至此，基于 TCP/IP 的 socket 网络编程就已经完成了。