**实验五：SOCKET通信**

1. 实验目的

1. 了解 TCP/IP 协议

2. 掌握 socket 编程

1. 实验环境

开发机环境

操作系统：ubuntu 12.04

交叉编译环境：arm-linux-gcc 4.3.2

6410板子内核源码：linux-3.0.1

目标板环境：OK6410-A linux-3.0.1

1. 实验原理
2. 计算机网络体系结构模式

所有的网络通信方式分为两种：线路交换和包交换。所谓的线路交换，就是指再传输时在发送端和接收端之间建立一个特定的线路连接，数据就可以在这条线路上传输。电话是采用的这种方式。计算机网络则采用的是包交换，数据的发送端将要传输的数据分割成块，而每个块经过适当的处理后形成一个数据包，包中有接收端的地址等必要信息，每个包单独传输。包中的数据并不是限定死的，只要保证数据的正确传输即可，具体应该定义哪些信息，则与使用的协议有关。

1. OSI标准

OSI标准是开放系统互联标准（Open Systerm Interconnection）即我们通常所说的网络互联的七层框架，他是1977年国际标准化组织提出的一种参考模型。值得注意的是，OSI并没有提供一个可以实现的方法，它不是一个标准而只是一个制定标准时使用的概念性的框架，更不是一个网络协议。

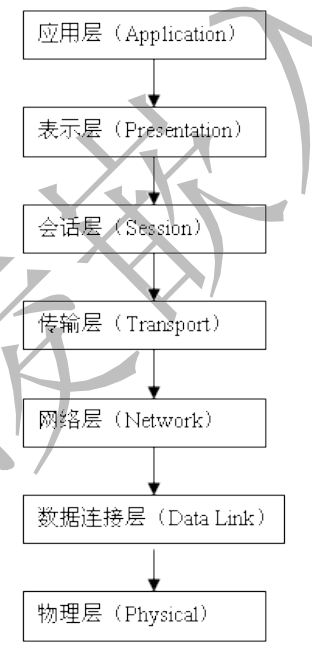
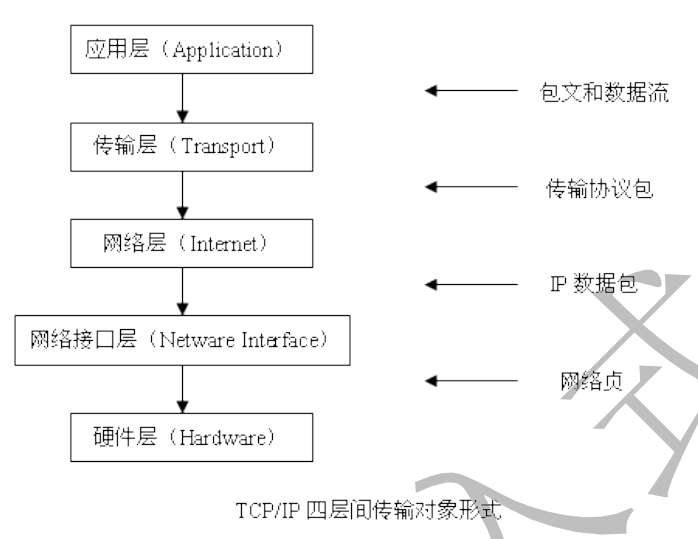


图1-网络互联的七层框架

1. TCP/IP协议

在实际应用中，最重要的是TCP/IP（Transport Control Protocol/Internet Protocol）协议，它是目前最流行的商业化的协议，相对于OSI，它是当前的工业标准或“事实的标准”，在1974年由Kahn提出的。它分为四个层次（从高到低）：应用层（与OSI的应用层对应），传输层（与OSI的传输层对应），互联层（与OSI的网络层对应），主机-网络层（与OSI的数据链路层和物理层应）。



1. 客户机与服务器

TCP/IP 允许程序员在两个应用程序之间建立通信并来回传递数据，提供一种对等通信，这种对等应用程序可以在同一台机器上，也可以在不同的机器上运行。尽管 TCP/IP 指明了数据是如何在一对正在通信的应用程序间传递的，但是他并没有规定对等的应用程序在什么时间进行交互以及为什么要进行交互，也没有规定程序员在一个分布式环境下应该如何组织这些应用程序。实践中，有一种组织的方法在使用 TCP/IP 中占据着主要地位，现在网络上的绝大多数的通信应用程序都使用这种机制。

客户机/服务器模式要求每个应用程序应有两部分组成：一个部分负责启动通信，另一个部分负责对他进行应答。他们通常运行在不同的主机上，分别被称为客户机和服务器。服务器是指能在网络上可提供服务的任何程序；客户机是指用户为了得到某种服务所需要运行的应用程序。一个服务器接受网络上客户机的请求，完成服务后将结果返回给客户机。

1. Socket编程相关函数
2. socket(建立连接)
3. bind(对socket定位)
4. listen(等待连接)
5. accept
6. connect(建立socket连线)
7. send()和recv()函数用于面向连接的socket上进行数据传输。
8. 实验代码
9. 客户端：client.c

*#include* <sys/types.h>

*#include* <sys/socket.h>

*#include* <stdio.h>

*#include* <netinet/in.h>

*#include* <arpa/inet.h>

*#include* <unistd.h>

*#include* <string.h>

*#include* <netdb.h>

*#include* <sys/ioctl.h>

*#include* <termios.h>

*#include* <stdlib.h>

*#include* <sys/stat.h>

*#include* <fcntl.h>

*#include* <signal.h>

*#include* <sys/time.h>

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

{

    pid\_t fd;

*const* char buff*[]* = "Hello! Welcome here gggggggggggg!\r\n"; *//定义要发送的数据缓冲区；*

    int sockfd, connsock;          *//定义一个socket套接字，用于通讯*

*struct* sockaddr\_in serveraddr; *//定义网络套接字地址结构*

*if* (argc != 2)

    {

        printf("Usage: echo ip地址");

        exit(0);

    }

    sockfd = socket(AF\_INET, SOCK\_STREAM, 0); *//创建一个套接字*

    bzero(&serveraddr, sizeof(serveraddr));

    serveraddr.sin\_family = AF\_INET;   *//指定使用的通讯协议族*

    serveraddr.sin\_port = htons(5000); *//指定要连接的服务器的端口*

    inet\_pton(AF\_INET, argv[1], &serveraddr.sin\_addr);

    connect(sockfd, (*struct* sockaddr \*)&serveraddr, sizeof(serveraddr)); *//连接服务器*

    send(sockfd, buff, sizeof(buff), 0);                                 *//向客户端发送数据*

    close(sockfd);                                                       *//关闭套接字*

*return* (0);

}

1. 服务器端：service.c

*#include* <sys/types.h>

*#include* <sys/socket.h>

*#include* <stdio.h>

*#include* <netinet/in.h>

*#include* <arpa/inet.h>

*#include* <unistd.h>

*#include* <string.h>

*#include* <netdb.h>

*#include* <sys/ioctl.h>

*#include* <termios.h>

*#include* <stdlib.h>

*#include* <sys/stat.h>

*#include* <fcntl.h>

*#include* <signal.h>

*#include* <sys/time.h>

void main()

{

    pid\_t fd;

    int listensock, connsock;

    char recvbuff[100];                                                           *//定义要接收的数据缓冲区*

*struct* sockaddr\_in serveraddr;                                                *//定义网络套接字地址结构*

    listensock = socket(AF\_INET, SOCK\_STREAM, 0);                                 *//创建一个套接字，用于监听*

    bzero(&serveraddr, sizeof(*struct* sockaddr));                                  *//地址结构清零*

    serveraddr.sin\_family = AF\_INET;                                              *//指定使用的通讯协议族*

    serveraddr.sin\_addr.s\_addr = htonl(INADDR\_ANY);                               *//指定接受任何连接*

    serveraddr.sin\_port = htons(5000);                                            *//指定监听的端口*

    bind(listensock, (*struct* sockaddr \*)&serveraddr, sizeof(*struct* sockaddr\_in)); *//给套接 口邦定地址*

    listen(listensock, 1024); *//开始监听*

    connsock = accept(listensock, (*struct* sockaddr \*)NULL, NULL);    *//建立通讯的套接字，accept函数，等待客户端程序使用connect函数的连接*

    recv(connsock, recvbuff, sizeof(recvbuff), 0); *//接收服务器的数据*

    printf("%s\n", recvbuff);                      *//打印接收到的数据*

    sleep(2);

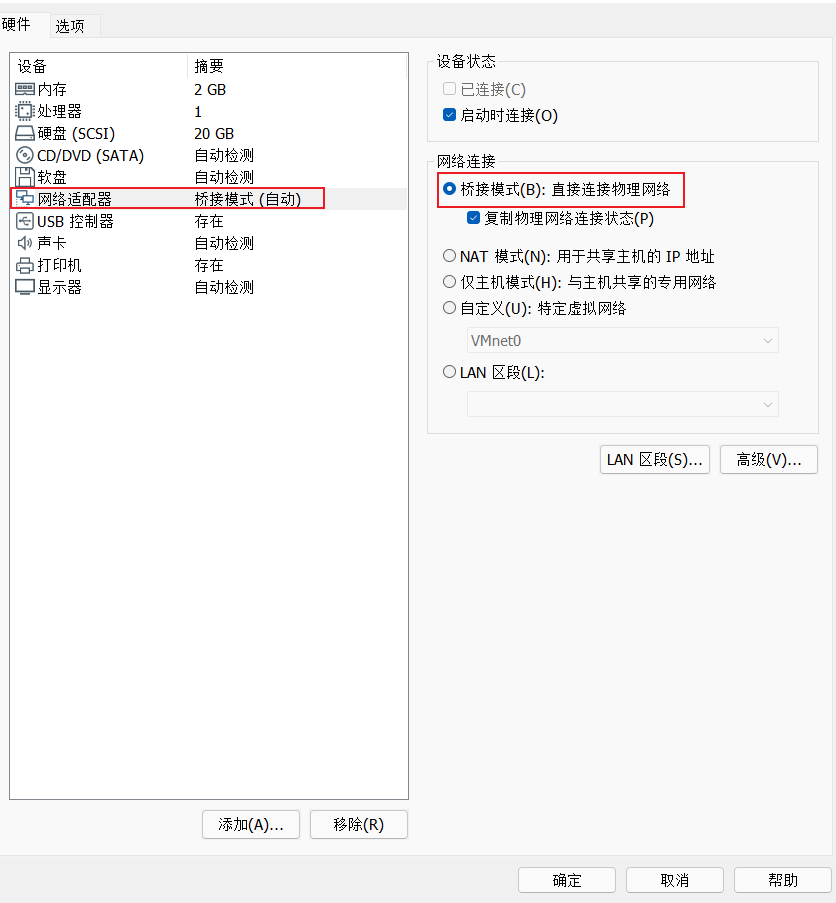
    close(connsock);   *//关闭通讯套接字*

    close(listensock); *//关闭监听套接字*

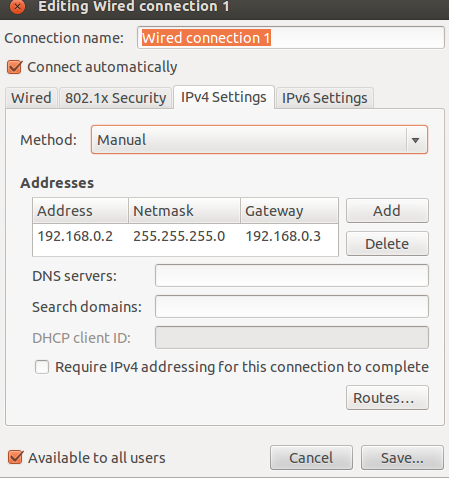
}

1. 实验步骤
2. 用网线连接开发板和PC机
3. 设置虚拟机为桥接模式



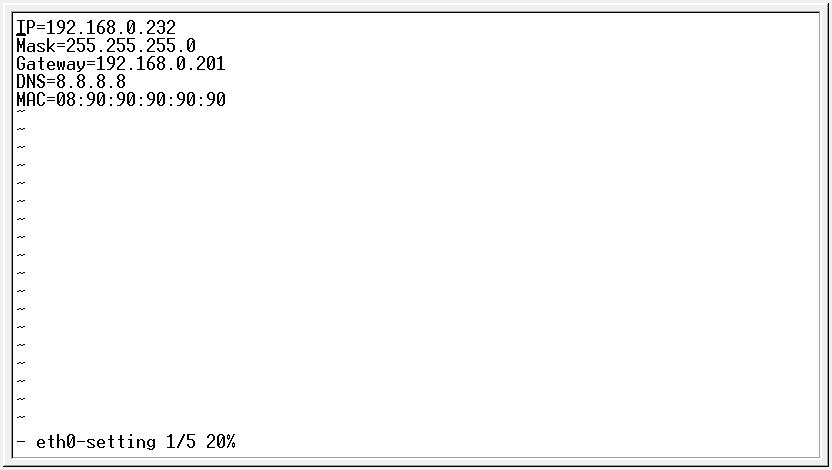


1. 更改ubuntu网络参数

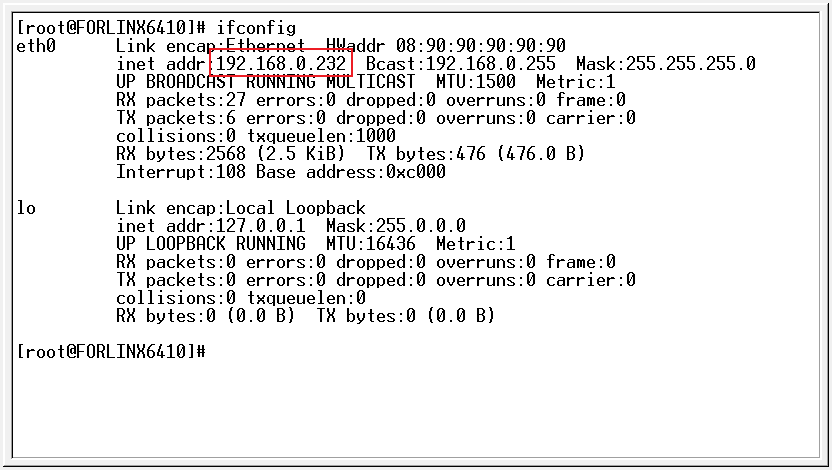


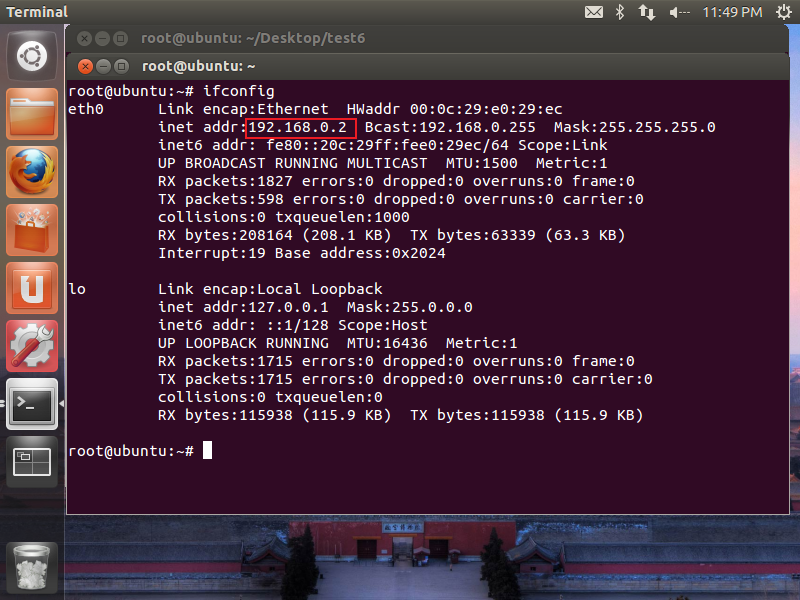
1. 设置开发板网络参数

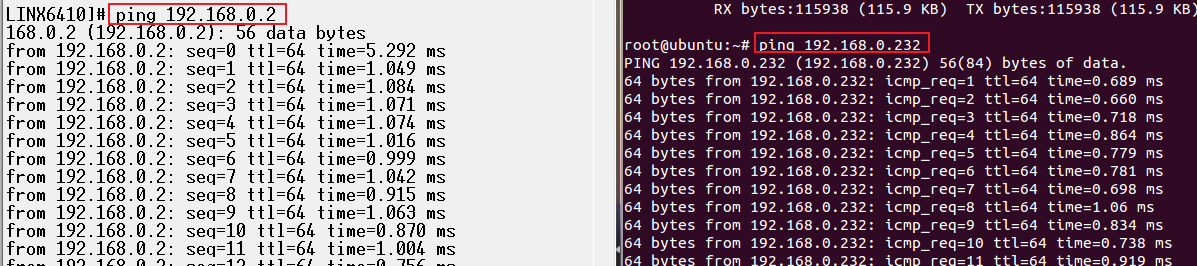
在开发板终端中执行 #vi \*\*\* etc/eth0-srtting,更改参数后保存退出



1. 查看开发板和ubuntu的IP地址，使两者可以连通







1. 编译及运行

在ubuntu中对实验代码进行编译，以ubuntu作为服务器，开发板作为客户端。

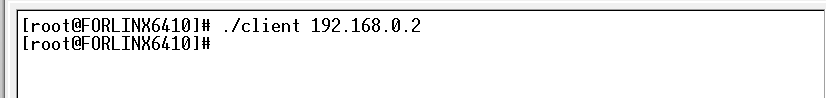
编译服务器代码：#gcc service.c -o service

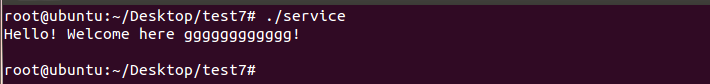
编译客户端代码：#arm-linux-gcc client.c -o client

1. 运行结果

在ubuntu中运行：#./service

在开发板中运行:#./client 192.168.0.2





客服端与PC端成功通信。

将开发板作为服务器，Ubuntu作为客户端，也成功通信。

1. 实验总结

本次实验主要是关于socket编程及协议的使用，在网络配置方面遇到了困难，通过查找资料，最终成功连接。了解了gcc和arm-linux-gcc编译的不同点和相同点.