

实验报告

课程：嵌入式系统A

第六次实验

**姓 名 凌智城**

**学 号 201806061211**

**专业班级 通信工程1803班**

**老 师 黄国兴**

**学 院 信息工程学院**

**提交日期**  **2021年6月9日**

实验10：以太网传输程序编写实验

1. 实验目的
2. 通过实验了解以太网通讯原理和驱动程序开发方法。
3. 通过实验了解 TCP 和 UDP 协议的功能和作用。
4. 通过实验了解基于 TCP/UDP 的 socket 编程。
5. 实验内容
6. 学习 KSZ8001L 网卡驱动程序。
7. 测试网卡功能，编写基于 TFP/UDP 协议网络聊天程序，并且能够接受键盘输入和彼此之间相互发数据。
8. 实验步骤

**步骤1：**硬件连接

虚拟机：打开虚拟机

接着查看串口号，通过putty软件使用串口通信方式连接实验箱，如下图所示：



图10-1 端口号查询

选择putty串口连接实验箱:

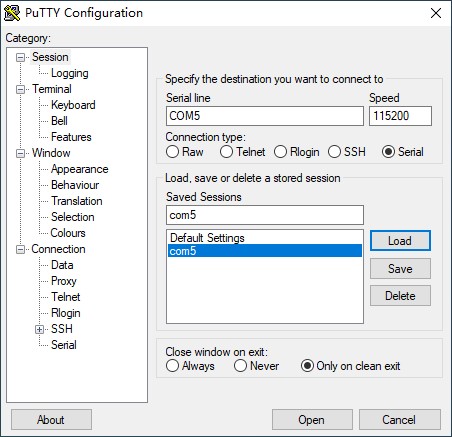


图10-2 putty串口连接配置

输入启动参数，接着启动内核

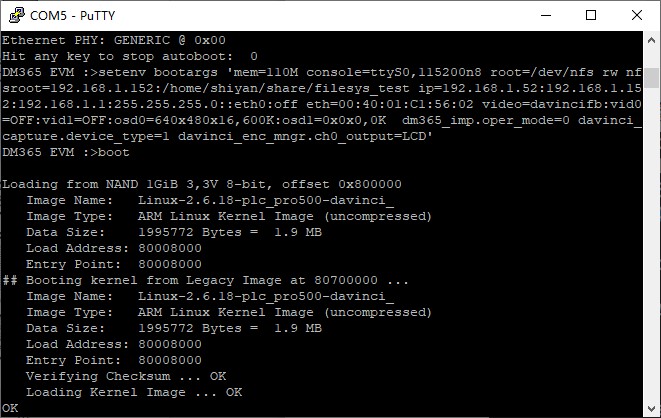


图10-3 输入启动参数启动

输入用户名root登录实验箱：

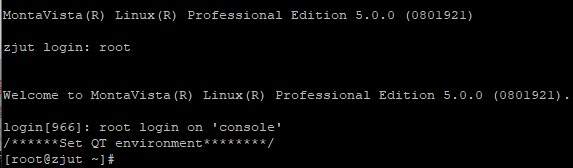


图10-4 登录实验箱

**步骤2：**测试网络连通性（在虚拟机/服务器上）

利用ping命令测试，输入命令：

|  |
| --- |
| $ ping 192.168.1.52 |
| (192.168.1.(50+x)为板子设定的ip，根据实际情况来，此时使用的为第二胎实验箱，x=2，故为52) |

并观察响应时间和丢包率，判断连接是否正常, 如果正常，说明 ARP ,IP , ICMP 协议正常：（使用“CTRL+c” 退出ping测试）

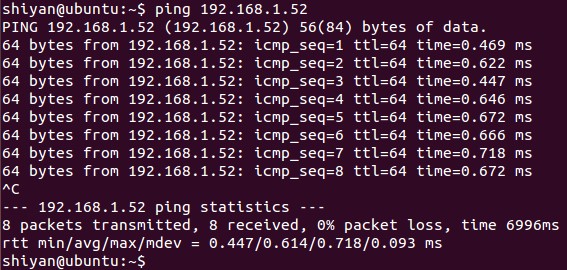


图10-5 ping命令

**步骤3：**基于TCP的socket编程

编写好 TCP 代码，包括服务器端和客户端代码。在用户home目录中创建ethenet目录，进入该目录创建tcpclient.c和tcpserver.c文件，代码在附录中。

|  |
| --- |
| $ mkdir ethenet  $ cd ethenet/  $ vim tcpserver.c  $ vim tcpclient.c |

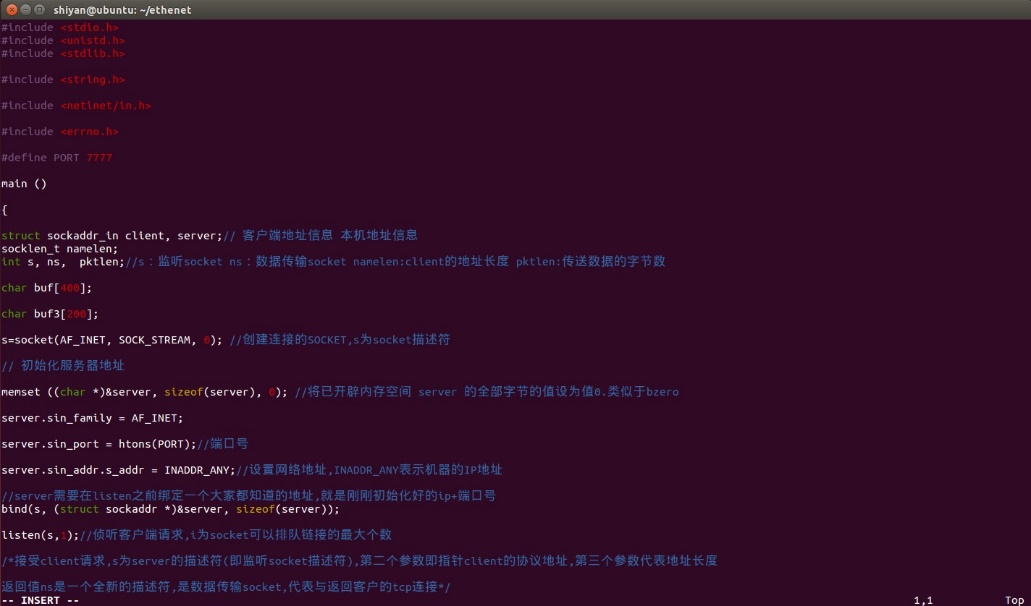


图10-6 tcpserver.c

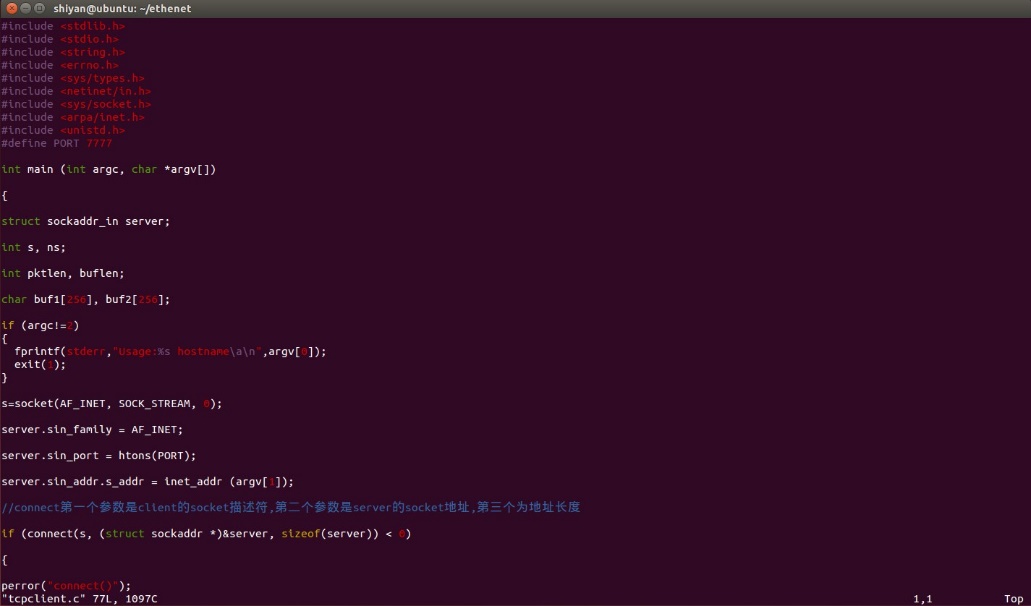


图10-6 tcpclient.c

编辑完毕客户端和服务器端程序后分别执行:wq指令保存

退回根目录运行source /etc/profile指令使环境变量生效

输入命令编译：

|  |
| --- |
| $ arm\_v5t\_le-gcc tcpclient.c -o tcpclient\_arm  $ gcc tcpserver.c -o tcpserver-gcc |

E:\大三下\嵌入式系统\Experiments\6th_20210609\Print screen\实验10：以太网传输程序编写实验\生成客户端代码和服务端代码.jpg

图10-7 生成TCP客户端代码和服务端代码

第一行命令将编译生成可以在实验箱上运行的tcp客户端代码，第二行编译生成可以在pc（虚拟机）或linux服务器上运行的代码，如果第二行改为如下所示，那么服务器代码也可以在实验箱上运行。

|  |
| --- |
| $ arm\_v5t\_le- gcc tcpserver.c -o tcpserver\_arm |

挂载文件系统：

|  |
| --- |
| mount -t nfs -o nolock 192.168.1.152:/home/shiyan/share/filesys\_test /mnt/mtd/ |

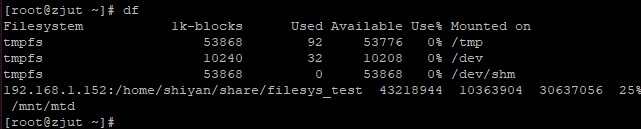


图10-8 挂载文件系统到实验箱

然 后 将 生 成 的 可 执 行 文 件 拷 贝 到 文 件 系 统 的/home/shiyan/share/filesys\_test/opt/dm365 下：

|  |
| --- |
| **服务器**：$ cp tcpclient\_arm /home/stx/filesys\_test/opt/dm365  **虚拟机**：$ cp tcpclient\_arm /home/shiyan/share/filesys\_test/opt/dm365 |

E:\大三下\嵌入式系统\Experiments\6th_20210609\Print screen\实验10：以太网传输程序编写实验\将tcpclient复制到挂载系统文件夹.jpg

图10-9 将tcpclient复制到挂载系统文件夹

在服务端运行./tcpserver-gcc代码等待客户端连接，客户端运行./tcpserver\_arm 192.168.1.152连接成功（192.168.1.152是服务端IP地址）。

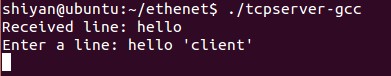


图10-10 tcp服务端成功接收客户端发送的hello

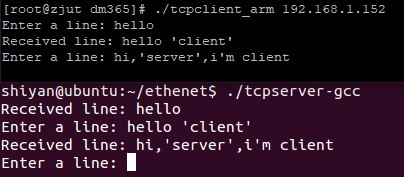


图10-10 tcp客户端成功接收服务端发送的hello ‘client’并且继续发送消息

**步骤4：**基于UDP的socket编程

编写好 TCP 代码，包括服务器端和客户端代码。在ethenet目录，进入该目录创建udpclient.c和udpserver.c文件，代码在附录中。

|  |
| --- |
| $ vim udpserver.c  $ vim udpclient.c |

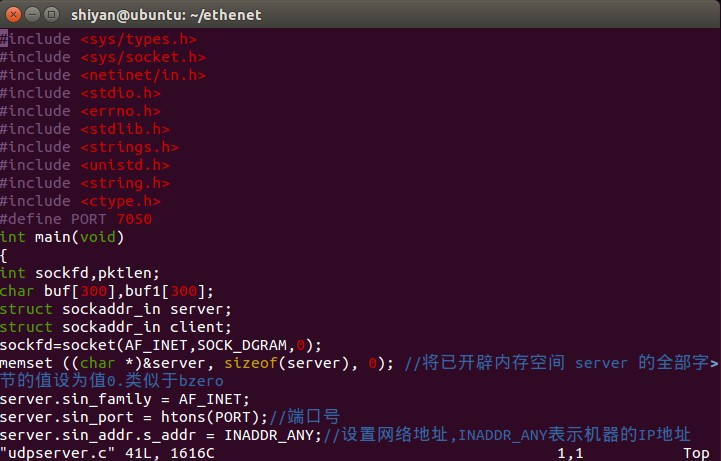


图10-11 udpserver.c



图10-11 udpclient.c

编辑完毕客户端和服务器端程序后分别执行:wq指令保存

退回根目录运行source /etc/profile指令使环境变量生效

输入命令编译：

|  |
| --- |
| $ arm\_v5t\_le-gcc udpclient.c -o udpclient\_arm  $ gcc udpserver.c -o udpserver-gcc |

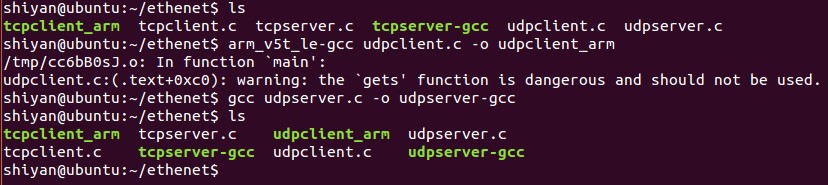


图10-12 生成UDP客户端代码和服务端代码

第一行命令将编译生成可以在实验箱上运行的udp客户端代码，第二行编译生成可以在pc（虚拟机）或linux服务器上运行的代码，如果第二行改为如下所示，那么服务器代码也可以在实验箱上运行。

|  |
| --- |
| $ arm\_v5t\_le- gcc udpserver.c -o udpserver\_arm |

(步骤三中已经挂载文件系统)然 后 将 生 成 的 可 执 行 文 件 拷 贝 到 文 件 系 统 的/home/shiyan/share/filesys\_test/opt/dm365 下：

|  |
| --- |
| **服务器**：$ cp udpclient\_arm /home/stx/filesys\_test/opt/dm365  **虚拟机**：$ cp udpclient\_arm /home/shiyan/share/filesys\_test/opt/dm365 |

E:\大三下\嵌入式系统\Experiments\6th_20210609\Print screen\实验10：以太网传输程序编写实验\将udpclient复制到挂载系统文件夹.jpg

图10-13 将udpclient复制到挂载系统文件夹

在服务端直接运行./udpserver-gcc指令等待客户端连接：

E:\大三下\嵌入式系统\Experiments\6th_20210609\Print screen\实验10：以太网传输程序编写实验\udp等待客户端连接.jpg

图10-14 udp等待客户端连接

在实验箱执行./udpclient\_arm 192.168.1.152命令连接成功：

E:\大三下\嵌入式系统\Experiments\6th_20210609\Print screen\实验10：以太网传输程序编写实验\udp客户端连接成功.jpg

图10-15 udp客户端连接成功

client向server发送：hello,’udpserver’，server成功接收

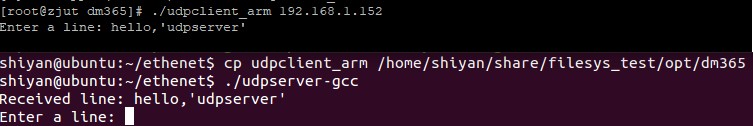


图10-16 client发送，server接收

server向client回复：hi,’udpclient’,I’m udpserver，client成功接收



图10-17 server发送，client接收

1. 心得与体会

通过这次以太网传输实验，对之前学习的TCP/UDP协议网络通信协议有了更加清晰的理解，参考已有TCP和UDP的服务端与客户端程序实现服务器和客户端简单的网络双机通信，接下来要进一步理解相应程序代码，对网络编程的基本原理和常用的函数充分理解，以为之后网络编程的继续学习打下良好基础。

1. 附录

tcpserver.c

|  |
| --- |
| #include <stdio.h>  #include <unistd.h>  #include <stdlib.h>  #include <string.h>  #include <netinet/in.h  #include <errno.h>  #define PORT 7777  main ()  {  struct sockaddr\_in client, server;// 客户端地址信息 本机地址信息  socklen\_t namelen;  int s, ns, pktlen;//s：监听socket ns：数据传输socket namelen:client的地址长度 pktlen:传送数据的字节数  char buf[400];  char buf3[200];  s=socket(AF\_INET, SOCK\_STREAM, 0); //创建连接的SOCKET,s为socket描述符  // 初始化服务器地址  memset ((char \*)&server, sizeof(server), 0); //将已开辟内存空间 server 的全部字节的值设为值0.类似于bzero  server.sin\_family = AF\_INET;  server.sin\_port = htons(PORT);//端口号  server.sin\_addr.s\_addr = INADDR\_ANY;//设置网络地址,INADDR\_ANY表示机器的IP地址  //server需要在listen之前绑定一个大家都知道的地址,就是刚刚初始化好的ip+端口号  bind(s, (struct sockaddr \*)&server, sizeof(server));  listen(s,1);//侦听客户端请求,i为socket可以排队链接的最大个数  /\*接受client请求,s为server的描述符(即监听socket描述符),第二个参数即指针client的协议地址,第三个参数代表地址长度  返回值ns是一个全新的描述符,是数据传输socket,代表与返回客户的tcp连接\*/  namelen = sizeof (client);  ns = accept (s, (struct sockaddr \*)&client, &namelen);  //开始进行网络I/O  for (;;) {  /\*recv接受client发送的数据,recv函数仅仅是copy数据，真正的接收数据是协议来完成的）， 第一个参数指定接收端套接字描述符；第二个参数指明一个缓冲区，该缓冲区用来存放recv函数接收到的数据；第三个参数指明buf的长度  recv函数返回其实际copy的字节数\*/  pktlen = recv (ns, buf, sizeof (buf), 0);  if (pktlen == 0)  break;  printf ("Received line: %s\n", buf);  printf ("Enter a line: ");  gets(buf3);  /\*并不是send把ns的发送缓冲中的数据传到连接的另一端的，而是协议传的，send仅仅是把buf中的数据copy到ns的发送缓冲区的剩余空间里  返回实际copy的字节数\*/  send (ns, buf3,sizeof(buf3), 0);  }  close(ns);  close(s);  } |

tcpclient.c

|  |
| --- |
| #include <stdlib.h>  #include <stdio.h>  #include <string.h>  #include <errno.h>  #include <sys/types.h>  #include <netinet/in.h>  #include <sys/socket.h>  #include <arpa/inet.h>  #include <unistd.h>  #define PORT 7777  int main (int argc, char \*argv[])  {  struct sockaddr\_in server;  int s, ns;  int pktlen, buflen;  char buf1[256], buf2[256];  if (argc!=2)  {  fprintf(stderr,"Usage:%s hostname\a\n",argv[0]);  exit(1);  }  s=socket(AF\_INET, SOCK\_STREAM, 0);  server.sin\_family = AF\_INET;  server.sin\_port = htons(PORT);  server.sin\_addr.s\_addr = inet\_addr (argv[1]);  //connect第一个参数是client的socket描述符,第二个参数是server的socket地址,第三个为地址长度  if (connect(s, (struct sockaddr \*)&server, sizeof(server)) < 0)  {  perror("connect()");  exit(1);  }  //进行网络I/O  for (;;) {  printf ("Enter a line: ");  gets (buf1);//从stdin流中读取字符串，直至接受到换行符  buflen = strlen (buf1);  if (buflen == 0)  break;  send(s, buf1, buflen + 1, 0);  recv(s, buf2, sizeof (buf2), 0);  printf("Received line: %s\n", buf2);  }  close(s);  return 0;  } |

udpserver.c

|  |
| --- |
| #include <sys/types.h>  #include <sys/socket.h>  #include <netinet/in.h>  #include <stdio.h>  #include <errno.h>  #include <stdlib.h>  #include <strings.h>  #include <unistd.h>  #include <string.h>  #include <ctype.h>  #define PORT 7050  int main(void)  {  int sockfd,pktlen;  char buf[300],buf1[300];  struct sockaddr\_in server;  struct sockaddr\_in client;  sockfd=socket(AF\_INET,SOCK\_DGRAM,0);  memset ((char \*)&server, sizeof(server), 0); //将已开辟内存空间 server 的全部字节的值设为值0.类似于bzero  server.sin\_family = AF\_INET;  server.sin\_port = htons(PORT);//端口号  server.sin\_addr.s\_addr = INADDR\_ANY;//设置网络地址,INADDR\_ANY表示机器的IP地址  bind(sockfd,(struct sockaddr \*)&server,sizeof(struct sockaddr\_in));  for (;;) {  /\*recv接受client发送的数据,recv函数仅仅是copy数据，真正的接收数据是协议来完成的）， 第一个参数指定接收端套接字描述符；第二个参数指明一个缓冲区，该缓冲区用来存放recv函数接收到的数据；第三个参数指明buf的长度  recv函数返回其实际copy的字节数\*/  socklen\_t len=sizeof(struct sockaddr\_in);  pktlen = recvfrom (sockfd, buf, sizeof (buf), 0,(struct sockaddr \*)&client, &len);  if (pktlen == 0)  break;  printf ("Received line: %s\n", buf);  printf ("Enter a line: ");  fgets(buf1,300,stdin);  /\*并不是send把ns的发送缓冲中的数据传到连接的另一端的，而是协议传的，send仅仅是把buf中的数据copy到ns的发送缓冲区的剩余空间里  返回实际copy的字节数\*/  sendto (sockfd, buf1,sizeof(buf1), 0,(struct sockaddr\*)&client,len);  }  close(sockfd);  } |

udpclient.c

|  |
| --- |
| #include <sys/types.h>  #include <sys/socket.h>  #include <netinet/in.h>  #include <errno.h>  #include <stdio.h>  #include <unistd.h>  #include <string.h>  #include <stdlib.h>  #include <arpa/inet.h>  #define MAX\_SIZE 1024  #define PORT 7050  #define HOST\_ADDR "192.168.1.51" //根据组队的服务器地址更改  int main(int argc,char \*\*argv)  {  int sockfd,buflen;  char buf1[300],buf2[300];  struct sockaddr\_in server,client;  socklen\_t client\_length=sizeof(client);  if(argc!=2)  {  fprintf(stderr,"Usage:%s hostname\a\n",argv[0]);  exit(1);  }  sockfd=socket(AF\_INET,SOCK\_DGRAM,0);  server.sin\_family=AF\_INET;  server.sin\_port = htons(PORT);  server.sin\_addr.s\_addr = inet\_addr(argv[1]);  for(;;)  {  printf ("Enter a line: ");  gets (buf1);//从stdin流中读取字符串，直至接受到换行符  buflen = strlen (buf1);  if (buflen == 0)  break;  sendto(sockfd, buf1, buflen + 1, 0,(struct sockaddr \*)&server,sizeof(server));  if(recvfrom(sockfd, buf2, sizeof (buf2), 0,(struct sockaddr \*)&client,&client\_length)==-1)  {  printf("recvfrom() error\n");  exit(1);  }  printf("Received line: %s\n", buf2);  }  close(sockfd);  return 0;  } |