**实训日志**

|  |  |  |
| --- | --- | --- |
| 学校：宝鸡文理学院 | 专业：电子科学与技术 | 学号：201691074036 |
| 日期；9月12日 | 星期：星期三 | 姓名：刘超飞 |
| 内容：（课程主要内容，代码/作业，收获，感悟，建议等）  课程主要内容：  网络编程  1.网络的发展史  2.网络协议模型  3.网络的相关概念  4.网络编程(Tcp C/S)  1.网络的发展史：  1969：ARP网  2.网络协议模型  1.ISO-OSI七层协议模型  应用层：提供应用层服务的(FTP\TFTP\NFS\HTTP\SSH\Telnet\SMTP\DNS\DHCP)  表示层：数据格式的表示和加密  会话层：建立会话的  传输层：定义传输数据的方式(TCP\UDP)  网络层：IP:主要用户组网和路由(IGMP\ICMP)  数据链路层：传输以太网帧(SLIP\PPP)  物理层：802.2/802.3 802.11  \*2.TCP/IP四层协议模型  应用层：  传输层：  网络层：  数据链接&物理接口层：  3.网络的相关概念：  1.socket：  >是一种进程间的通信机制(接口)  >socket是一个函数  >socket还是一种文件类型  2.进程：是程序动态执行的一次过程  3.程序：编译出来的可执行文件    A主机 B主机  微信 hello 微信  4.IP地址：用于标明网络主机的一个编号，32bit，通常用点分十进制的形式表示。  在一个局域网内，IP地址是唯一的。  IP地址通常与子网掩码配套使用  IPV4:  网络类型 网络号 主机号  5.子网掩码：是一个32bit的二进制数，用1来掩网络号，用0来掩主机号。    192.168.1.1  A 192.168.1.22  B 192.168.1.33  255.255.255.0  A 1100 0000 1010 1000 0000 0001 0001 0110  B 1100 0000 1010 1000 0000 0001 0010 0001  1111 1111 1111 1111 1111 1111 0000 0000  6.端口号：是用于寻找目的主机上的目的进程的。  是一个16bit的一个正整数  0~65535  在自己写服务的时候，端口号给5000以后  7.大小端  0x12 34 56 78  高位 低位  低地址 78 12  56 34  34 58  12 78  高地址  int x = 0x12345678；  char \*p = (char \*)&x;  p: \*p:  p+1: \*(p+1):  p+2: \*(p+2):  p+3: \*(p+3):  丹尼斯·里奇  肯·汤普森  小端存储：低地址存低位，高地址存高位  大端存储：低地址存高位，高地址存低位  4：网络编程模型：  C/S模型：  客户端服务器模型  B/S模型：  浏览器服务器模型  5.TCP/IP C/S模型的搭建：  服务器搭建的流程：  <1>socket  <2>bind  <3>listen  <4>accept  <5>send and recv  <6>close  客户端的搭建：  <1>socket  <2>connect  <3>send and recv  <4>close  socket：创建一个通信端口  /\*需要添加的头文件\*/  #include <sys/types.h> /\* See NOTES \*/  #include <sys/socket.h>  /\*  \*函数名：socket  \*函数功能：创建一个通信端口  \*函数参数：  \* int domain：使用的协议族(我们使用的是AF\_INET)  \* int type：要创建的socket的类型（我们使用的是tcp流式套接字：SOCK\_STREAM）  \* int protocol：使用的协议(默认为0)  \*函数返回值：成功返回创建的socket的文件描述符，失败返回-1  \*/  int socket(int domain, int type, int protocol);  bind：给socket绑定一个名字  /\*需要添加的头文件\*/  #include <sys/types.h> /\* See NOTES \*/  #include <sys/socket.h>  /\*  \*函数名：bind  \*函数功能：给socket绑定IP和端口号  \*函数参数：  \* int sockfd：被创建出来的socket的文件描述符  \* const struct sockaddr \*addr：addr是一个 struct sockaddr类型的一个指针，它指向被绑定的内容  \* socklen\_t addrlen：是struct sockaddr这个数据类型所占的字节数  \*函数返回值：成功返回0，失败返回-1  \*/  int bind(int sockfd, const struct sockaddr \*addr, socklen\_t addrlen);  struct sockaddr {  sa\_family\_t sa\_family;  char sa\_data[14];  }  struct sockaddr\_in {  sa\_family\_t sin\_family; /\* address family: AF\_INET \*/  in\_port\_t sin\_port; /\* port in network byte order \*/  struct in\_addr sin\_addr; /\* internet address \*/  };  /\* Internet address. \*/  struct in\_addr {  uint32\_t s\_addr; /\* address in network byte order \*/  };    主机字节序：小端存储  网络字节序：大端存储  如何将主机字节序转换位网络字节序：  /\*头文件\*/  #include <arpa/inet.h>  /\*  \*将主机字节序的四字节的数字转换成网络字节序的四字节的数字  \*/  uint32\_t htonl(uint32\_t hostlong);  /\*  \*将主机字节序的二字节的数字转换成网络字节序的二字节的数字  \*/  uint16\_t htons(uint16\_t hostshort);  /\* Address to accept any incoming messages. \*/  #define INADDR\_ANY ((unsigned long int) 0x00000000)  listen：监听客户端的链接请求  /\*需要添加的头文件\*/  #include <sys/types.h> /\* See NOTES \*/  #include <sys/socket.h>  /\*  \*函数名：listen  \*函数功能：监听客户端的链接请求  \*函数参数：  \* int sockfd：被创建出来的socket的文件描述符  \* int backlog：监听队列的大小  \*函数返回值：成功返回0，失败返回-1  \*/  int listen(int sockfd, int backlog);  accept：接受客户端的连接请求  #include <sys/types.h> /\* See NOTES \*/  #include <sys/socket.h>  /\*  \*函数名：accept  \*函数功能：接受客户端的连接请求，并返回一个通信套接字  \*函数参数：  \* int sockfd：监听套接字  \* struct sockaddr \*addr：用于存储客户端的地址信息  \* socklen\_t \*addrlen:地址空间的长度  \*函数返回值：成功返回通信套接字的文件描述符,失败返回-1  \*/  int accept(int sockfd, struct sockaddr \*addr, socklen\_t \*addrlen);  send：  /\*头文件\*/  #include <sys/types.h>  #include <sys/socket.h>  /\*  \*函数名：send  \*函数功能：给对端发送一个buf(数据)  \*函数参数：  \* int sockfd:通信套接字的文件描述符  \* const void \*buf:想要发送的内容的首地址  \* size\_t len:想要发送的字节数  \* int flags:默认为0  \*函数返回值：成功返回发送的字节数，失败返回-1  \*/  ssize\_t send(int sockfd, const void \*buf, size\_t len, int flags);  recv:  /\*头文件\*/  #include <sys/types.h>  #include <sys/socket.h>  /\*  \*函数名：recv  \*函数功能：从对端接收一个buf(数据)  \*函数参数：  \* int sockfd:通信套接字的文件描述符  \* const void \*buf:存储接收的内容的首地址  \* size\_t len:想要接收的字节数  \* int flags:默认为0  \*函数返回值：成功返回接收的字节数，失败返回-1。当返回值为0时，代表对端执行了一个有序关闭。  \*/  ssize\_t recv(int sockfd, void \*buf, size\_t len, int flags);  connect：  /\*头文件\*/  #include <sys/types.h> /\* See NOTES \*/  #include <sys/socket.h>  /\*  \*函数名：connect  \*函数功能：发送一个socket的连接请求  \*函数参数：  \* int sockfd：创建客户端的套接字  \* const struct sockaddr \*addr:存储服务器的IP和端口号的结构的地址  \* socklen\_t addrlen:结构的长度  \*函数的返回值：成功返回0，失败返回-1.  \*/  int connect(int sockfd, const struct sockaddr \*addr, socklen\_t addrlen);    bind的时候：IP和端口号字节序的问题：  htons/htonl  192.168.2.12  in\_addr\_t inet\_addr(const char \*cp);  收获：  今天学习的是网络编程，先是实现网络初始化，这部分比较简单只要完成必要的步骤就可以实现。第二步是实现客户端和服务器信息的链接和收发问题，这部分就比较难了。通过最后不懈的努力终于完成了chat 程序的编写，通过努力获得的成功很爽。 | | |