中国科学院软件研究所

课程文档

|  |  |  |
| --- | --- | --- |
| 课程名称 | ： | 《openEuler应用编程技术》 |
| 教师姓名 | ： |  |
| 提交时间 | ： |  |

中国科学院软件研究所智能软件研究中心制

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 课程  名称 | 《openEuler应用编程技术》 | | 章 | | 第七章 | | | 课程类型 |
| 节 | | 第二节 | | | 授课(√ )  实训( ) |
| 名称 | 网络编程基础（下） | | | | | | |
| 教师 |  | 课时 | | | | 1课时 | | |
| 参考  资料 | 1. Unix环境高级编程（第2版），[美] 史蒂文斯，拉戈著尤晋元，张亚英，戚正伟译，人民邮电出版社 | | | | | | | |
| 教学  目的  要求 | 掌握：  基本套接字操作  TCP套接字编程  UDP套接字编程  基本套接字操作中的各个网络链接函数使用  TCP套接字编程异常情况处理 | | | | | | | |
| 教学  重点  难点 |  | | | PPT页面 | | | 时间分配 | |
| **教学重点** | | | | | | | |
| 基本套接字操作 | | | 3~18页 | | | 5分钟 | |
| TCP套接字编程 | | | 19~40页 | | | 12分钟 | |
| UDP套接字编程 | | | 41~57页 | | | 1分钟 | |
| **教学难点** | | | | | | | |
| 基本套接字操作中的各个网络链接函数使用 | | | 3~17页 | | | 5分钟 | |
| TCP套接字编程异常情况处理 | | | 37～40页 | | | 5分钟 | |

|  |  |  |
| --- | --- | --- |
| 教学方法 | | |
| 本授课以课堂讲授为主，与课堂实时练习方式相结合 | | |
| 教学内容 | 操作演示 | 知识点 |
| **PPT第1页:**  接下来讲第七章Linux socket网络编程的第2节。 |  | 目录 |
| **PPT第2页:**  这一节主要介绍TCP和UDP相关的一些编程。 |  | 大纲 |
| **PPT第3页:**  这里给出了一系列的函数。  主要是完成客户端-服务器端模型的基本套接字操作函数。  我们先来了解一下客户端-服务器端模型的基本流程是怎么样的？  对于服务器端：  第一步：创建一个socket套接字，使用socket()函数。  第二步：绑定IP地址、端口等信息到socket上，使用bind()函数。  第三步：设置套接字为监听模式，允许的最大连接数，使用listen()函数。  第四步：接收客户端请求，建立连接，使用accept()函数  第五步：收发数据，使用send()/recv()函数，或read()/write()函数。  第六步：最后关闭网络连接。  对于客户端：  第一步：创建一个socket，使用socket()函数。  第二步：设置要连接的对方的IP地址和端口等属性。  第三步：连接服务器，使用connect()函数。  第四步：收发数据使用send()/recv()函数，或read()/write()函数。  第五步：最后关闭网络连接。  接下来具体看每个函数的用法。 |  | 基本套接字操作 |
| **PPT第4页:**  首先看第一个函数socket。  int socket(int domain, int type, int protocol);  第一个参数：domain常用的值有三个，  AF\_INET，是大多数用来产生socket的协议，使用TCP或UDP来传输，用IPv4的地址；  AF\_INET6 与上面类似，不过是来用IPv6的地址  AF\_UNIX 本地协议，使用在Unix和Linux系统上，一般都是当客户端和服务器在同一台机器上的时候使用。  第二个参数：type常用的有三种SOCK\_STREAM、SOCK\_DGRAM、SOCK\_RAW。  SOCK\_STREAM 这个协议是按照顺序的、可靠的、数据完整的基于字节流的连接。这是一个使用最多的socket类  型，这个socket是使用TCP来进行传输。  SOCK\_DGRAM 这个协议是无连接的、固定长度的传输调用。该协议是不可靠的，使用UDP来进行它的连接。  SOCK\_RAW 这个socket类型提供单一的网络访问，这个socket类型使用ICMP公共协议。（ping、traceroute使  用该协议）。  第三个参数：protocol为0 ，默认协议。  socket()打开一个网络通讯端口，  如果成功的话，就像open()一样返回一个文件描述符，应用程序可以像读写文件一样用read/write在网络上收发数据，  如果出错则返回-1。对于IPv4，domain参数指定为AF\_INET。对于TCP协议，type参数指定为SOCK\_STREAM，表示面向流的传输协议。如果是UDP协议，则type参数指定为SOCK\_DGRAM，表示面向数据报的传输协议。protocol参数的介绍从略，指定为0即可。 |  | socket |
| **PPT第5页:**  前面socket函数创建了一个套接字，接下来通过bind函数绑定一个指定的固定的Ip和端口号。看下bind函数的具体的参数。  第一个参数：sockfd为socket文件描述符。  第二个参数：addr构造出IP地址加端口号。  第三个参数：addrlen为第二个参数addr的长度。  服务器程序所监听的网络地址和端口号通常是固定不变的，客户端程序得知服务器程序的地址和端口号后就可以向服务器发起连接，因此服务器需要调用bind绑定一个固定的网络地址和端口号。  bind()的作用是将参数sockfd和addr绑定在一起，使sockfd这个用于网络通讯的文件描述符监听addr所描述的地址和端口号。前面讲过，struct sockaddr \*是一个通用指针类型，addr参数实际上可以接受多种协议的sockaddr结构体，而它们的长度各不相同，所以需要第三个参数addrlen指定结构体的长度。    首先将整个结构体清零，然后设置地址类型为AF\_INET，网络地址为INADDR\_ANY，这个宏表示本地的任意IP地址，因为服务器可能有多个网卡，每个网卡也可能绑定多个IP地址，这样设置可以在所有的IP地址上监听，直到与某个客户端建立了连接时才确定下来到底用哪个IP地址，端口号为8000。 |  | bind |
| **PPT第6页:**  继续看下Listen函数。  第一个参数sockfd表示socket的文件描述符。  第二个参数backlog表示排队建立3次握手队列和刚刚建立3次握手队列的链接数和。  断点  典型的服务器程序可以同时服务于多个客户端，当有客户端发起连接时，服务器调用的accept()返回并接受这个连接，如果有大量的客户端发起连接而服务器来不及处理，尚未accept的客户端就处于连接等待状态，listen()声明sockfd处于监听状态，并且最多允许有backlog个客户端处于连接待状态，如果接收到更多的连接请求就忽略。listen()成功返回0，失败返回-1。 | **操作演示：**  1.演示目的：查看系统默认的backlog。  2.打开终端。  3.在终端界面敲：   1. cat /proc/sys/net/ipv4/tcp\_max\_syn\_backlog。   查看系统默认backlog为128。 | Listen |
| **PPT第7页:**  下面是Accept函数。  第一个参数sockfd表示socket文件描述符。  第二个参数addr是一个传出参数，返回链接客户端地址信息，含IP地址和端口号。  第三个参数addrlen是传入传出参数，传入sizeof(addr)大小 ，函数返回时返回真正接收到地址结构体的大小。  三方握手完成后，服务器调用accept()接受连接，如果服务器调用accept()时还没有客户端的连接请求，就阻塞等待直到有客户端连接上来。addr是一个传出参数，accept()返回时传出客户端的地址和端口号。addrlen参数是一个传入传出参数（value-resultargument），传入的是调用者提供的缓冲区addr的长度以避免缓冲区溢出问题，传出的是客户端地址结构体的实际长度（有可能没有占满调用者提供的缓冲区）。如果给addr参数传NULL，表示不关心客户端的地址。 |  | accept |
| **PPT第8页:**  接下来是connect函数。  第一个参数sockfd表示socket文件描述符。  第二个参数addr是传入参数，指定服务器端地址信息，含IP地址和端口号。  第三个参数是传入参数，传入sizeof(addr)大小。  客户端需要调用connect()连接服务器，connect和bind的参数形式一致，区别在于bind的参数是自己的地址，而connect的参数是对方的地址。connect()成功返回0，出错返回-1。 |  | connect |
| **PPT第9页:**  通过前面的connect连接成功以后就可以正常的收发数据。发送数据的话可以用write。往sockfd文件描述符写buf里面的数据。  write的返回值大于0,表示写了部分或者是全部的数据. 这样我们用一个while循环来不停的写入，但是循环过程中的buf参数和nbyte参数得由我们来更新。也就是说，网络写函数是不负责将全部数据写完之后在返回的。  返回的值小于0,此时出现了错误.我们要根据错误类型来处理。如果错误为EINTR表示在写的时候出现了中断错误。 如果为EPIPE表示网络连接出现了问题(对方已经关闭了连接)。 |  | write |
| PPT第10页: 和write对应的就是read，将sockfd文件描述符的数据读到buffer当中。从网络读，根据不同的传输层协议和内核缓存机制，返回值可能小于请求的字节数。参数count是请求读取的字节数，读上来的数据保存在缓冲区buf中，同时文件的当前读写位置向后移。 |  | read |
| PPT第11页: Send也可以用于发送数据。 |  | send |
| PPT第12页: recv和send函数提供了和read和write差不多的功能.不过它们提供了第四个参数来控制读写操作.  前面的三个参数和read,write一样,第四个参数可以是0就等同于read、write函数。 |  | recv |
| PPT第13页: close用于关闭指定的套接字。 |  | close |
| PPT第14页: 根据accept返回的sockfd，得到临时的端口号。 |  | getsockname |
| PPT第15页:通过函数名，我们可以晓得，setsockopt 就是设这sock的一些选项，那这些参数都是什么意义呢？ 获取或者设置与某个套接字关联的选项。选项可能存在于多层协议中，它们总会出现在最上面的套接字层。当操作套接字选项时，选项位于的层和选项的名称必须给出。为了操作套接字层的选项，应该 将层的值指定为SOL\_SOCKET。为了操作其它层的选项，控制选项的合适协议号必须给出。  参数：    sock：将要被设置或者获取选项的套接字。 level：选项所在的协议层。 optname：需要访问的选项名。 optval：对于getsockopt()，指向返回选项值的缓冲。对于setsockopt()，指向包含新选项值的缓冲。 optlen：对于getsockopt()，作为入口参数时，选项值的最大长度。作为出口参数时，选项值的实际长度。对于setsockopt()，现选项的长度。 |  | Setsockopt |
| PPT第16页 getpeername根据accpet返回的sockfd，这个sockfd跟客户端进行通信，那么客户端远端的端口号是多少呢？  可以通过getpeername得到远端链接的端口号。Sockfd是服务器端的fd，可以通过函数得到远端客户端的信息。accept返回的时候已经保存了客户端的IP地址和端口号。 |  | getpeername |
| PPT第17页: 基本套接字操作涉及的函数如这张图所示，前面一直在逐个讲解的。 |  |  |
| PPT第18页: 上面介绍了一些通用的函数，在下面的套接字编程中肯定是需要用到的，但是怎么用，如何把它们串起来，实现什么功能，其实还是有一定距离的。 |  | 大纲 |
| PPT第19页:   现在我们可以结合前面讲的函数来看下这幅图。  服务器端首先创建一条socket，bind是为了让自己的IP地址和端口号绑定前面创建的socket，然后listen设置服务器在同一时刻能最多允许多少个连接请求，accept等待客户端连接，accept返回了以后，意味着有一个客户端连接上来,accept会返回一条新的文件描述符，在和客户端通信的时候，我们是用这条新的文件描述符进行通信，于是服务器端就去read这条新的文件描述符，服务器端如果读到客户端发送过来的数据，然后进行处理，处理完以后回送到客户端，然后就是read、write处理，最后客户端关闭连接，注意服务器端的这个close，是close掉accept刚才返回的那条socket。  客户端想要和服务器端通信，也是要先创建一个socket，然后知道服务器端的IP地址和端口号connect建立连接，连接成功以后，客户端可以向建立的socket进行write操作，服务器读到客户端发送的信息以后，进一步进行处理。 |  | TCP  套接字编程 |
| PPT第20页: 我们来总结一下，对于服务器来说，就是这几步。  首先创建一个socket套接字，用socket函数，然后绑定IP地址和端口号等信息到socket上，用bind函数，再把前面的创建的套接字设置为监听的模式，设置一下允许的最大的连接数用listen函数，然后再接收客户端的请求，建立连接用accept函数，这个时候会返回新的一个套接字，然后就可以收发数据，最后关闭网络连接。 |  | TCP  套接字编程 |
| PPT第21页: 对于客户端是创建一个socket，使用socket函数，客户端会自动被分配一个临时的端口号，设置要连接的对方的IP地址和端口等属性，然后再使用connect()函数连接服务器，连接成功之后就可以通过send()/recv()，或read()/write()函数直接收发数据：最后关闭网络连接。 |  | TCP  套接字编程 |
| PPT第22页: 接下来是TCP服务器程序的模板，先通过（标号1）socket函数创建一个套接字，并返回一个socket文件描述符sockfd，AF\_INET和SOCK\_STREAM共同标志使用的是TCP，0是表示默认的协议。  创建一个新的套接字之后，（标号2）然后绑定IP地址和端口号等信息到socket上，用bind函数，再把前面的创建的套接字设置为监听的模式，设置一下允许的最大的连接数用listen函数。  再进入一个循环等待客户端连接的过程，（标号3）然后再接收客户端的请求，建立连接用accept函数，第二个参数和第三个参数都为NULL，表示不关心客户端的IP地址等信息，这个时候会返回新的一个套接字，然后就可以收发数据，最后（标号4）关闭网络连接。 |  | TCP  服务器程序模板 |
| PPT第23页: 接下来是客户端程序模板，（标号1）创建一个socket，使用socket函数，客户端会自动被分配一个临时的端口号，设置要连接的对方的IP地址和端口等属性，（标号2）然后再使用connect()函数连接服务器，连接成功之后就可以通过（标号3）send()/recv()，或read()/write()函数直接收发数据，（标号4）最后关闭网络连接。 |  | TCP  客户端程序模板 |
| PPT第24页 首先看下Server端的总步骤：  第一步：程序初始化  第二步：持续监听一个固定的端口  第三步：收到Client的连接后建立一个socket连接  第四步：与Client进行通信和信息处理  第五步：接收Client通过socket连接发送来的数据，进行相应处理并返回处理结果，如BBS Server  第六步：通过socket连接向Client发送信息 |  |  |
| PPT第25页 接下来看具体的流程：  程序流程一：  首先通过socket函数创建一个新的socket套接字，这个新的套接字放在sockfd里面，通过AF\_INET和SOCK\_STREAM指定为TCP协议。  程序流程二：  进一步填充自身的地址信息sockaddr\_in结构，  将sockaddr\_in里面的三个字段填写完整，  第一个字段sin\_family赋值为AF\_INET；  第二个字段sin\_port通过htons函数转化为网络字节序的端口号。  第三个字段sin\_addr.s\_addr赋值为INADDR\_ANY，可以为任意IP地址。  Sockaddr\_in结构体中其余部分置0. |  | server程序 |
| PPT第26页 程序流程三：  绑定前面设置好的地址相关的结构体信息。  因为前面设置的sockaddr\_in结构体，但是bind函数的第二个参数需要的是struct sockaddr\*结构体，所以这里注意一定要强制转换为struct sockaddr\*结构体类型；  程序流程四：  监听前面socket创建的那个套接字文件描述符sockfd，并设置最多允许连接的数目 BACKLOG为10。  程序流程五：  定义一个客户端的地址信息，通过accept函数与客户端进行连接，这个时候会返回新的一个套接字，然后就可以收发数据。 |  | server程序 |
| PPT第27页 程序流程六：  产生新的进程处理读写socket，  子进程进行发送Hello，world；父进程关闭accept返回的新文件描述符new\_fd。  程序流程七：  转程序流程五，继续等待其他Client的连接并处理 |  | server程序 |
| PPT第28页 接下来是具体的server程序的源代码  包含了必须的头文件和定义的一些结构体变量。 |  | server程序 |
| PPT第29页 通过socket创建套接字，并返回一个套接字文件描述符sockfd，然后绑定IP地址和端口号等信息到socket上，用bind函数，再把前面的创建的套接字设置为监听的模式，设置一下允许的最大的连接数用listen函数。 |  | server程序 |
| PPT第30页 然后再接收客户端的请求，建立连接用accept函数，这个时候会返回新的一个套接字，然后就可以收发数据，最后关闭网络连接。 |  | server程序 |
| PPT第31页 接下来是Client端的总步骤：  第一步：程序初始化  第二步：连接到某个Server上，建立socket连接  第三步：与Server进行通信和信息处理  第四步：接收Server通过socket连接发送来的数据，进行相应处理  第五步：通过socket连接向Server发送请求信息  第六步：通信结束后中断与Client的连接 |  | Client程序 |
| PPT第32页 接下来是客户端程序的总流程：  程序流程一：  首先通过socket函数创建一个新的socket套接字，返回一个socket文件描述符sockfd，通过AF\_INET和SOCK\_STREAM指定是TCP协议。  程序流程二：  填写客户端的地址信息sockaddr\_in结构，首先定义一个hostent结构体和sockaddr\_in结构体，通过gethostbyname函数返回对应于给定[主机名](http://baike.baidu.com/view/1617349.htm)whitehouse.gov的包含主机名字和地址信息的hostent结构[指针](http://baike.baidu.com/view/159417.htm)，再设置sockaddr\_in的三个字段的信息，  第一个字段sin\_family赋值为AF\_INET；  第二个字段sin\_port通过htons函数转化为网络字节序的端口号。  第三个字段sin\_addr.s\_addr赋值为对方的IP地址。  Sockaddr\_in结构体中其余部分置0. |  | Client程序 |
| PPT第33页 程序流程三：  通过connect函数连接服务器端口。  程序流程四：  读写socket。  程序流程五：  关闭socket。 |  | Client程序 |
| PPT第34页 接下来是具体的client程序的源代码  包含了必须的头文件和定义的一些结构体变量。 |  | Client程序 |
| PPT第35页 通过gethostbyname获取客户端的地址信息等。  并且填充好客户端的sockaddr\_in结构体信息。 |  | Client程序 |
| PPT第36页 再通过connect函数连接服务器；  连接成功后，进行读写socket；  最后关闭socket。 |  | Client程序 |
| PPT第37页: TCP客户端/服务器端程序主要常见有以下三种异常情况，  第一种服务器主机崩溃。  第二种服务器主机崩溃后重启  第三种服务器主机关机。 |  | 异常情况 |
| PPT第38页: 第一种情况是服务器主机崩溃的情况：  当服务器主机崩溃后，服务器端不会向客户端发任何数据，但是这时客户端并不知道，客户端还是会不断的向服务器端发送数据，然后客户端就会阻塞在从服务器端读数据。  因为客户端给服务器端发送数据，但是一直没有收到服务器的ACK，所以客户端会不断的重传数据。  当客户端重传达到了12次，放弃发送数据，此时，客户端会接收到ETIMEDOUT错误。  还有可能通过网络层的ICMP的协议，发送ICMP包时，被告知服务器主机已经崩溃。  当然也可以通过设置套接字选项更改TCP持续重传等待的超时时间，这样可以提前知道服务器已经崩溃了。 |  | 服务器主机崩溃 |
| PPT第39页: 第二种情况是服务器主机崩溃后重启。  如果客户端在服务器主机崩溃并重启了的这段时间内没有给服务器端发送消息的话，客户端是不知道服务器重启过的，  由于服务器重启后丢失了之前的连接信息，这时，如果客户端给服务器发送数据，这时，服务器会给客户端发送一个RST，这样，当客户端TCP收到RST才知道服务器是崩溃后重启的。  同样，也可以设置套接字选项，即是客户端不向服务器端发送消息，客户端也能收到服务器端的崩溃情况。 |  | 服务器主机崩溃重启 |
| PPT第40页: 第三种情况是服务器主机关机。  当Linux系统关机的时候，由init进程给所有运行的进程发SIGTERM信号，如果服务器程序忽略SIGTERM信号，则init进程会等待一段固定的时间，给所有还在运行的程序发SIGKILL信号，服务器将由SIGKILL信号终止，其终止时，所有打开的文件描述符被关闭，然后就会向客户发送FIN。  客户收到FIN后，就知道服务器将关机。 |  | 服务器主机关机 |
| PPT第41页: 服务器端首先创建一条socket，bind是为了让自己的IP地址和端口号绑定前面创建的socket，然后listen设置服务器在同一时刻能最多允许多少个连接请求，accept等待客户端连接，accept返回了以后，意味着有一个客户端连接上来,accept会返回一条新的文件描述符，在和客户端通信的时候，我们是用这条新的文件描述符进行通信，于是服务器端就去read这条新的文件描述符，服务器端如果读到客户端发送过来的数据，然后进行处理，处理完以后回送到客户端，然后就是read、write处理，最后客户端关闭连接，注意服务器端的这个close，是close掉accept刚才返回的那条socket。 客户端想要和服务器端通信，也是要先创建一个socket，然后知道服务器端的IP地址和端口号connect建立连接，连接成功以后，客户端可以向建立的socket进行write操作，服务器读到客户端发送的信息以后，进一步进行处理。 |  |  |
| PPT第42页: 接下来介绍的是UDP套接字的编程。UDP与TCP有什么不同呢？  TCP是面向连接的，可靠的，有流量控制，拥塞控制等机制。 UDP是与TCP相对应的协议。它是面向无连接的协议，它不与对方建立连接，而是直接就把数据包发送过去！ |  | 大纲 |
| PPT第43页: 我们一起来看下UDP套接字的工作流程，UDP和TCP不一样的地方是服务器端listen，客户端没有connect。  但是相关的功能在哪里实现了呢？主要是由服务器端用到的recvfrom函数，客户端用到的sendto函数去实现的。Recvfrom函数和sendto函数的最后二个参数会给出相应的客户端的IP地址信息等，服务器端没有listen，是通过recvfrom函数一直阻塞直到有客户端sendto给服务器端，然后再通过sendto函数和recvfrom函数进行数据应答。 |  | UDP套接字的工作流程 |
| PPT第44页: 在UDP套接字编程当中，服务器端的步骤是： 首先创建一个socket使用socket()函数，然后通过bind()函数绑定IP地址、端口等信息到socket上，再通过recvfrom()函数循环接收数据：处理客户端请求，再通过sendto()函数发送信息回客户端，最后通过close函数关闭网络连接。 |  | UDP套接字的工作流程 |
| PPT第45页: 客户端的就更简单，  首先通过 socket()函数创建一个socket：  然后 通过sendto()函数连接到服务器端后，就可以发送数据：再通过recvfrom()函数接收数据应答；最后关闭网络连接。 |  | UDP套接字的工作流程 |
| PPT第46页:Sendto函数和以前讲过的send函数是类似的，不同的是多了最后两个参数，用来指定发送到的目的端的具体的地址信息。 |  | Sendto |
| PPT第47页:Recvfrom函数和以前讲过的recv函数是类似的，不同的是多了最后的两个参数，用来指定接收端的具体的地址信息。 |  | recvfrom |
| PPT第48页: 这里有一个变量的特殊用法值-结果参数，也即是传入传出参数，现在进行说明一下。  bind函数相当于是一个用户进程到内核的值传递。  recvfrom函数相当于是一个内核到用户进程的结果传递，  写进来的时候是一个值，返回来后是一个指针，大家需要注意的是recvfrom的最后一个参数是一个指针。  当bind函数被调用的时候，结构大小是一个值，这样可以是内核再写此结构的时候不至于越界；  当recvfom函数返回时，结构大小又是一个结果，为了告诉进程内核在此结构中确切存储了多少信息。Recvfrom函数的这个最后一个参数就是一个传入传出参数。 |  | 值-结果参数说明 |
| PPT第49页:接下来看下UDP服务器程序的模板，首先通过socket函数创建一个新的套接字，这里通过AF\_INET和SOCK\_DGRAM指定是UDP协议，最后一个参数默认情况下指定为0，然后通过bind()函数绑定IP地址、端口等信息到socket上，再通过recvfrom()函数循环接收数据：处理客户端请求，再通过sendto()函数发送信息回客户端，最后通过close函数关闭网络连接。 |  | UDP 服务器程序模板 |
| PPT第50页: 对于客户端，首先通过 socket()函数创建一个socket：  然后通过sendto()函数连接到服务器端后，就可以发送数据：再通过recvfrom()函数接收数据应答；最后关闭网络连接。 |  | UDP 客户端程序模板 |
| PPT第51页:接下来是一个UDP协议的具体的client-server的例子。 首先看下server端的代码，这里给出了必要的头文件和构造UDP通信的套接字，使用了socket函数创建一个新的套接字。 |  | UDP |
| PPT第52页: 前面创建了一个套接字后，再通过bind函数绑定该服务器端，然后通过recvfrom函数接收客户端发送过来的数据，再通过toupper函数处理数据，将小写转大写。再通过sendto函数把处理后的数据发送给客户端，最后关闭网络连接。 |  | UDP |
| PPT第53页:然后看下client端的代码，客户端同样也是先通过socket函数创建一个新的套接字。 |  | UDP |
| PPT第54页:然后循环的从客户端输入数据存在buf当中，然后通过sendto函数发送到服务器端，服务器端进行处理以后，客户端再通过recvfrom函数接收服务器端发回来的数据放在buf当中，再通过write函数输出到屏幕上，最后关闭网络连接。 |  | UDP |
| PPT第55页: | **操作演示：**  1.演示目的：掌握简单UDP客户端-服务器端程序的编写。  2.打开终端。  3.在终端界面敲：   1. ./server 2. ./client   运行server端程序后，显示服务器端处理等待接收连接的状态，  运行cllient端程序后，客户端输入hello，hello经过服务器端处理后，发送回客户端，可以发现下一行出现大写的HELLO，并且在服务器端打印出了临时的端口号为59888。这就是一个简单的UDP客户端-服务器端程序。 | UDP |
| PPT第56页:由TCP和UDP的工作流程图可以看出TCP和UDP的异同点，TCP协议在通信之前是需要连接的，UDP是不需要的。 TCP和UDP都需要通过socket建立套接字，但是TCP接下来需要listen，UDP却不需要listen这个函数，因为UDP是在recvfrom函数和sendto函数去指定接收端地址信息和发送端的地址信息。  针对客户端，TCP是需要通过connect建立连接的,而UDP是不需要的，因为UDP是面向无连接的。  最后都需要关闭。 |  | TCP 与  UDP的比较 |
| PPT第57页 谢谢&问答 |  | 结尾 |
| 知识点框图 | | |
| 结构图 | | |