之后回顾的时候通过关键字去英文版中查找

1.1

Even though the client and server communicate using an application protocol, the

transport layers communicate using TCP. Note that the actual flflow of information

between the client and server goes down the protocol stack on one side, across the net

work, and up the protocol stack on the other side. Also note that the client and server

are typically user processes, while the TCP and IP protocols are normally part of the

1.11

ANSI C invented the size\_t datatype, which is used, for example, as the argument

to malloc (the number of bytes to allocate), and as the third argument to

read and

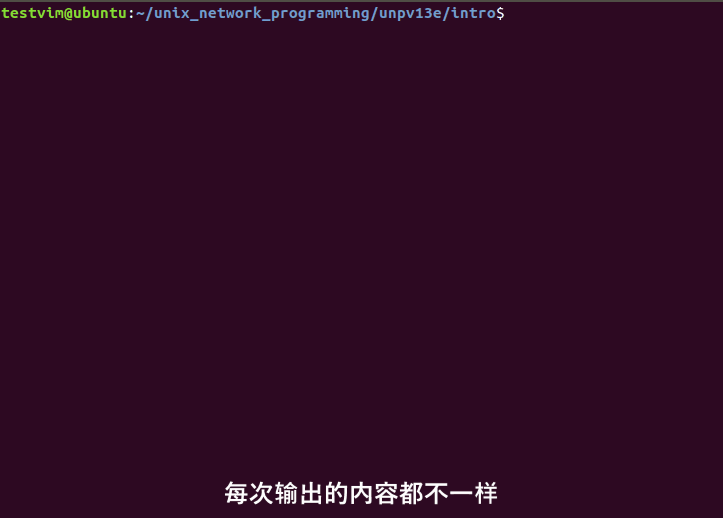
write (the number of bytes to read or write). On a 32-bit system, size\_t

is a 32-bit

value, but on a 64-bit system, it must be a 64-b

第一章练习：

1.5，为什么改变之后的代码每次运行客户端程序输出都不相同，不知道是fputs函数的原因，还是read函数的原因？



上图是一个GIF图片。

2.5 下面这段话给出了一种解决TCP传输数据发生阻塞时的解决方案，但是我没看懂。SCTP提供的阻塞解决方案是通过一个客户端机器和多个服务端机器建立连接来实现的，而TCP的这种解决方案不是。

Similar robustness can be obtained from TCP with help from routing protocols. For example,

BGP connections within a domain (iBGP) often use addresses that are assigned to a virtual

interface within the router as the endpoints of the TCP connection. The

不明白MSS option的作用是什么？

MSS option. With this option, the TCP sending the SYN announces its maximum segment size, the maximum amount of data that it is willing to accept in each TCP segment, on this connection. The sending TCP uses the receiver ’s MSS value as the maximum size of a segment that it sends. We will see how to fetch and set this TCP option with the TCP\_MAXSEG socket option (Section 7.9).

2.6

一个Unix进程的关闭是会关闭所有和他相关的描述符还是，将Unix系统中的描述符都关闭？

The sending of each FIN occurs when a socket is closed. We indicated that the

application calls close for this to happen, but realize that when a Unix process terminates, either voluntarily (calling exit or having the main function return) or involuntarily (receiving a signal that terminates the process), all open descriptors are closed, which will also cause a FIN to be sent on any TCP connection that is still open.

有些不明白Figure 2.4中的simultaneous open 和 simultaneous close是什么意思？

a simultaneous open (when both ends send SYNs at about the same time and the SYNs cross in the network) and a simultaneous close (when both ends send FINs at the same time).

2.8、

we will discuss the one-to-one and one-to-many interface styles in Section 9.2

2.11、

不是很明白，为什么MTU至少为68字节，可以保证IPv4包含最大的头部，以及最小的切片。

The minimum link MTU for IPv4 is 68 bytes. This permits a maximum-sized IPv4 header (20 bytes of fifixed header, 30 bytes of options) and minimum-sized fragment (the fragment offset is in units of 8 bytes).

关于MSS的说明，为什么使用IPv6的时候一次允许传输的最大数据量不需要减掉IPv6的报头长度。

MSS option is a 16-bit fifield, limiting the value to

65,535. This is fifine for IPv4, since the maximum amount of TCP data in an IPv4

datagram is 65,495 (65,535 minus the 20-byte IPv4 header and minus the 20-byte

TCP header). But with the IPv6 jumbo payload option, a different technique is

used (RFC 2675 [Borman, Deering, and Hinden 1999]). First, the maximum

amount of TCP data in an IPv6 datagram without the jumbo payload option is

65,515 (65,535 minus the 20-byte TCP header). Therefore, the MSS value of

65,535 is considered a special case that designates ‘‘infifinity.’’ This value is used

only if the jumbo payload option is being used, which requires an MTU that

exceeds 65,535. If TCP is using the jumbo payload option and receives an MSS

announcement of 65,535 from the peer, the limit on the datagram sizes that it

sends is just the interface MTU. If this turns out to be too large (i.e., there is a

link in the path with a smaller MTU), then path MTU discovery will determine

the smaller value.

文章并没有详细说明SCTP如何请求一个分片点的。

SCTP keeps a fragmentation point based on the smallest path MTU found to all

the peer’s addresses. This smallest MTU size is used to split large user messages

into smaller pieces that can be sent in one IP datagram. The SCTP\_MAXSEG

socket option can inflfluence this value, allowing the user to request a smaller

fragmentation point.

2.12、

如何使用telnet执行守护进程inetd提供的daytime等服务？

Often these services are provided by the inetd daemon on Unix hosts (Section 13.5). These standard services provide an easy testing facility using the standard Telnet client. For example, the following tests both the daytime and echo servers:

解决

3.2、

为什么由sin\_len字段可以简化变长socket地址结构的处理？

Having a length fifield simplififies the handling of variable-length socket address structures.

并不知道路由器套接字是怎么使用sin\_len字段的，需要看完18章，可能才会知道。

Even if the length fifield is present, we need never set it and need never examine it, unless we are dealing with routing sockets (Chapter 18)

不是很明白sin\_zero成员的作用是什么？

The sin\_zero member is unused, bu

不是很明白结构体的强制转化是一个什么样的机制？

bind(sockfd, (struct sockaddr \*) &serv, sizeof(serv));

回答：经过查阅资料发现，结构体是不能够进行强制转化的，只有结构体指针可以进行强制转化。

要想理解下面的话，就得理解64位处理器是如何对64位边界值进行访问的。

The members in this structure are ordered so that if the sockaddr\_in6 structure is 64-bit aligned, so is the 128-bit sin6\_addr member. On some 64-bit processors, data accesses of 64-bit values are optimized if stored on a 64-bit boundary.

3.7、

没明白Figure 3.12中给出的inet\_ntop函数中的结构体指针转字符指针是怎么转的？等之后看了c和指针之后看能不能解决。

const u\_char \*p = (const u\_char \*) addrptr;

3.8、

为什么下面的函数通过将返回值使用静态空间的形式可以改写成不可重入的函数，为什么这样就会线程安全？

char \*sock\_ntop(const struct sockaddr \*sockaddr, socklen\_t addrlen);

看完Section 11.18可能会有答案。

3.9、

不是很明白为什么第一个readline函数不好，而第二个更好，感觉有点儿看不懂代码，c语言学的不好？

Note that our readline function calls the system’s

3.10、

TCP提供的字节流是存在缺陷的，但是我不知道为什么会存在这种缺陷，以及本文作者处理这种缺陷时使用的readline函数（Section 3.9）为什么能够处理这种缺陷，c语言程序还是太差了。

TCP sockets provide a byte stream to an application: There are no record markers.

4.2、

不明白socket函数每个参数所表示的含义，族、socket类型、协议，以及为什么socket需要这三个参数？

int socket(int family, int type, int protocol);

4.3、

书中展示了三种connect函数执行失败的情况，其中有两种情况个人实操没出来。

connect error: Connection timed out

connect error: No route to host

4.4、

我觉得我得了解一下，底层const和顶层const了。

Indeed, it cannot return this value since the second argument to bind has the const qualififier.

查一查getsockname函数的用法。

此章节最后一个地方没有理解，就是将bind函数绑定通配符的方法应用于为多个组织提供web服务的主机，这么提供有什么好处，是怎样的一个模型结构，书中并没有详细指出。之后看了Section 14.2看能不能理解。

A common example of a process binding a non-wildcard IP address to a socket is a host that provides Web servers to multiple organizations (Section 14.2 of TCPv3). First, each organization has its own domain name, such as www.organization.com. Next, each organization’s domain name maps into a different IP address, but typically on the same subnet. For example, if the subnet is 198.69.10, the first organization’s IP address could be 198.69.10.128, the next 198.69.10.129, and so on. All these IP addresses are then aliased onto a single network interface (using the alias option of the ifconfig command on 4.4BSD, for example) so that the IP layer will accept incoming datagrams destined for any of the aliased addresses. Finally, one copy of the HTTP server is started for each organization and each copy binds only the IP address for that organization.

4.5、

测试一下getenv函数。

getenv

4.6、

什么是回环地址？查询一下！！

loopback address (127.0.0.1)

4.7、

不明白fork函数是如何同时给子进程和父进程返回值的，需要测试一下；还有就是fork函数会拷贝进程自身是指把代码复制一遍然后从头执行吗？我觉得不是从头到尾执行，而是从fork处开始执行。

If you have never seen this function before, the hard part in understanding fork is that it is called once but it returns twice. It returns once in the calling process (called the parent) with a return value that is the process ID of the newly created process (the child). It also returns once in the child, with a return value of 0. Hence, the return value tells the process whether it is the parent or the child

不知道为什么存储在硬盘上的可执行程序的文件能够运行的唯一方式是调用exec函数族中的函数。

The only way in which an executable program fifile on disk can be executed by Unix is for an existing process to call one of the six exec functions.

不明白六个exec函数他们的参数之间有什么关联？

The differences in the six exec functions are: (a) whether the program fifile to execute is specifified by a fifilename or a pathname; (b) whether the arguments to the new program are listed one by one or referenced through an array of pointers; and (c) whether the environment of the calling process is passed to the new program or whether a new environment is specifified.

4.10、

关于getsockname函数的说明都进行测试。

These two functions are required for the following reasons:

测试一下下面的获取socket的地址族的代码。

The sockfd\_to\_family function shown in Figure 4.19 returns the address family of a socket.

5.2、

有些忘了accept函数的功能是什么了？

Accept(listenfd, (SA \*) &cliaddr, &clilen);

5.7、

子进程结束，会给父进程发送一个SIGCHLD signal，没有对这个信号进行处理，子进程变成了僵尸进程，不知道为什么？

Finally, the SIGCHLD signal is sent to the parent when the server child termi

nates. This occurs in this example, but we do not catch the signal in our code,

5.8、

不是很懂函数指针的相关说明。

void (\*signal(int signo, void (\*func)(int)))(int);

已解决

没看明白

We can set the default disposition for a signal by setting its disposition to

SIG\_DFL. The default is normally to terminate a process on receipt of a signal, with certain signals also generating a core image of the process in its current

有些没看懂signal函数中为什么要将sigaction结构中的成员sa\_mask设置成empty set

POSIX allows us to specify a set of signals that will be blocked when our signal handler is called. Any signal that is blocked cannot be delivered to a process. We set the sa\_mask member to the empty set, which means that no additional signals will be blocked while our signal handler is running. POSIX guarantees that the signal being caught is always blocked while its handler is executing.

没有看懂为什么signal函数要对SIGALRM进行特殊的处理？

(The reason for making a special case for SIGALRM is that the purpose of generating this signal is normally to place a timeout on an I/O operation, as we will show in Section 14.2, in which case, we want the blocked system call to be interrupted by the signal.

为什么要将旧行为作为signal函数的返回值？

We call sigaction and then return the old action for the signal as the return value of the signal function.

这之后的内容有些看不明白

POSIX Signal Semantics

5.9、

不是很明白为什么信号处理函数中的return可以中断系统调用？

As part of the coding conventions used in this text, we always code an explicit

return in our signal handlers (Figure 5.7), even though falling off the end of the function does the same thing for a function returning void. When reading the code, the unnecessary return statement acts as a reminder that the return may interrupt a system call.

5.11、

本节中只是大概的说明了一下当accept返回之前连接停止该如何进行处理，没有讲述具体的代码和效果，根据本节最后的说明，本文将在section 16.6中进行介绍。

Connection Abor t before accept Returns

5.12、

有些没看懂本节后半部分的内容，即如何让客户端接收RST分节？将代码中的Readline函数之前增加了sleep(120)函数，也没有收到服务端发送的RST，使用tcpdump也没有收到相关的包，不知道是怎么回事？

另外整理一下tcp什么时候发送RST信号。

If the readline happens before the RST is received, as we’ve shown in our example, the result is an unexpected EOF in the client. But if the RST arrives fifirst, the result is an ECONNRESET (‘‘Connection reset by peer’’) error return from readline.

试试看等到Section 6.4中是否能够解决这个问题。

5.16、

不是很清楚在服务端主机正常关机之后如何在客户端使用select和poll进行服务端程序终止的检测。

As stated there, we must use the select or poll function in our client to have the client detect the termination of the server process as soon as it occurs.

Exercise 5.2

没看懂题目是什么意思？

What happens with our echo client/server if we run the client and redirect standard input to a binary fifile?

Exercise 5.7

这个题目不知道怎么做？

What happens in Figure 5.15 if the IP address of the server host that is specifified by the client in its call to connect is the IP address associated with the rightmost datalink on the server, instead of the IP address associated with the leftmost datalink on the server?

6.2、

什么是低潮限度？

low-water mark

低潮限度是否可以理解为是一个阈值，如果数据没达到这个阈值就不读，如果缓冲区没达到这个阈值就不能写入。

关于一个socket可以被读的四个条件、可以被写的四个条件以及发生异常的条件，书上说的太过于笼统，有些不明白？

A socket is ready for reading if any of the following four conditions is true:

A socket is ready for writing if any of the following four conditions is true:

A socket has an exception condition pending if there is out-of-band data for the socket or the socket is still at the out-of-band mark. (We will describe out-ofband data in Chapter 24.)书中会在24章对带外数据进行介绍。

没有理解：UDP socket的低潮限度小于发送缓冲区的大小时，那么该UDP socket总是处于可以被写入的状态什么意思？

As long as the send low-water mark for a UDP socket is less than the send buffer size (which should always be the default relationship), the UDP socket is always writable, since a connection is not required.

已解决：突然懂了，只要空闲的发送缓冲区大小大于低潮限度，就可以被写入。

6.9、

不是很懂pselect最后一个参数是什么意思？有什么用？

This allows the program to disable the delivery of certain signals, test some global variables that are set by the handlers for these now-disabled signals, and then call pselect, telling it to reset the signal mask.

之后18.5小结后会进一步讨论。

E6.2、  
目前对阻塞式socket以及非阻塞socket还不是很了解，应该和第七章的socket选项有关。

When describing the conditions for which select returns ‘‘writable’’ in Section 6.3, why did we need the qualififier that the socket had to be nonblocking for a write operation to return a positive value?

7.4、

不是很清楚什么是广播？

SO\_BROADCAST Socket Option

什么是环形缓冲区？

circular buffer

7.10、

7.10 SCTP Socket Options感觉现在看没什么意义，不能通过实践来加深理解，所以暂且跳过

# E7.2、为什么getsockopt函数输出的连接后的mss值与tcpdump获取的TCP包中的mss值不一致？

E7.4、不知道为什么，按照答案的说法没有出现EADDRINUSE的错误？

E7.5、E7.6、怎么测试都不行，不知道是sock程序本身的问题还是其他的问题？

E7.7、学不会。

E8.4、8.6不是很会做。

9.2、

SCTP preserves message boundaries; thus, application-layer message bound

aries are not required. For example, an application protocol based on TCP might do a write() system call to write a two-byte message length fifield x, followed by a write() system call that writes x bytes of data. However, if this is done with SCTP, the receiving SCTP will receive two separate messages (i.e., the read call will return twice: once with a two-byte message, and then again with an x byte message).

9.3、

什么是动态地址功能？

If an endpoint supports the dynamic address feature

9.14、Notififications可以之后遇到了再看看，没有例子，很难真正理解。

10.2、

不是很清楚SCTP中的流有什么作用？为什么要增加流数？使用sctp\_get\_no\_strms函数有什么用？When a message arrives, the server checks the stream\_increment flflag to see if it should increment the stream number. If the flflag is set (no arguments were passed on the command line), the server increments the stream number of the message. If that number grows larger than or equal to the maximum streams, which is obtained by calling our internal function call sctp\_get\_no\_strms, the server resets the stream to 0. The function sctp\_get\_no\_strms is not shown. It uses the SCTP\_STATUS SCTP socket option discussed in Section 7.10 to fifind the number of streams negotiated.

E10.5、这个题目没法弄，因为sctp增加流数的代码没有调通。

E10.7、这个题目还真的不知道。

Why can a server get away with not tracking the associations it has open? Is there any danger in not tracking associations?

11.2、

不是很懂DNS中CNAME这个资源记录有什么作用？

for ‘‘canonical name.’’ A common use is to assign

CNAME records for common services, such as ftp and www. If people

use these service names instead of the actual hostnames, it is transparent

when a service is moved to another host. For example, the following

could be CNAMEs for our host linux:

11.4、

不是很懂in-addr.arpa是什么意思？

In terms of the DNS, gethostbyaddr queries a name server for a PTR record in the in-addr.arpa domain.

E11.11、这个练习感觉要用多线程或者多进程，不然如何在getnameinfo函数执行的过程中计算时间呢？

getnameinfo can take a long time (up to 80 seconds) to return an error if a hostname cannot be found for an IP address.

13.4、不知道为什么要把描述符0、1、2导入黑洞文件中？

open("/dev/null", O\_RDONLY);

30 open("/dev/null", O\_RDWR);

31 open("/dev/null", O\_RDWR)

E12.1、不知道这道题的目的是什么？

Start an IPv6 FTP client on a dual-stack host running IPv4 and IPv6. Connect to an IPv4 FTP server, make sure the client is in ‘‘active’’ mode (perhaps issuing the passive command to turn off ‘‘passive’’ mode),

E12.2、不知道是不是操作正确？

Write a program that requires one command-line argument that is an IPv4 dotted-decimal address.

14.8、

不是很明白对一个套接字描述符操作为什么输入函数后面不能接输出函数，输出函数后面不能接输入函数？

TCP and UDP sockets are full-duplex. Standard I/O streams can also be full

duplex: we just open the stream with a type of r+, which means read-write.

14.9、

不是很明白图14.18中的代码if (n == 0 || (isfile && n == kev[i].data))

是什么意思？

如何将标准IO设置为无缓冲。

E14.2、

目前还没有办法做这道题。

In Figure 14.1, we print a warning if the process already has an alarm timer set. Modify the function to reset this alarm for the process after the connect, before the function returns.

15.4、

有些不是很懂对于一个域套接字，路径名存在并且是一个套接字，但没有打开的套接字描述符与路径名相关联是一个什么情况？

the pathname exists and is a socket, but no open socket descriptor is associated with the pathname;

不是很懂“与连接 Unix 域套接字相关联的权限测试与调用 open 以对路径名进行只写访问一样”这句话的意思。

The permission testing associated with the connect of a Unix domain socket is the same as if open had been called for write-only access to the pathname.

15.7、

不明白在检查recvfrom返回的msghdr结构中的cmsghdr是否存在时，为什么要加上长度的判断？

cmptr->cmsg\_len == CMSG\_LEN(sizeof(int))

15.8、

我的ubuntu系统中没有cmsgcred这个结构体所以我没法进行UNIX书中的代码测试，如果要进行测试可以尝试更换一个freebsd系统，这个系统中好像集成了这个结构体。

Starting the server and then running the client one time in another window produces the following output from the server:

16.2、

关于标准IO和IO多路复用并存时的隐患问题的还需要再读一读第六章。