Unix 网络编程教程

（请对照示例代码相互印证）

一、关于 TCP/IP 协议

TCP/IP 协议属于网络协议栈中的传输层和网络层，其中 TCP、UDP 传输协议属于传输层，IP 网络协议属于网络层。

在发送端，来自应用层的数据包先经过网络层 TCP/UDP 协议的封装，再送至网络层由 IP 进行封装；而在接收端，接收的数据报先经过 IP 协议进行“解封装”过程，再由 TCP/UDP 再次“解封装”，最后得到发送的数据。

1.1 TCP

TCP 进行数据传输的双方在进行通信之前需要建立“专线”的连接，也就是“三次握手”的过程。TCP 将对接收到的每一个数据报返回一个应答信号，发送方根据是否收到应答信号来决定是否重发数据报，以此来保证数据完整的传输。

1.2 UDP

UDP 数据直接进行发送而不建立连接，接收方也不对接受的数据报产生应答信号，不能对丢失的数据报进行重发。UDP 传输简单而高效，实时性好，但安全性不能得到保证，根据不同的应用，应当考虑选择 TCP 或者 UDP。

1.3 IP

网路上的数据报需要根据自身携带的 IP 信息决定发往的目的地以及路径的选择。一个 IP 分为网际标识段和主机标识段，数据报根据 IP 地址一层一层选择路由。IP 协议为单方面传输，不需要事先连接，但准确的送达需要正确的地址和畅通有效的可达路径。

二、socket 套接字

Socket 套接字就是为 TCP/UDP 对数据报的“封装”、建立连接以及上下层通信提供了操作接口，套接字本身可以如同普通文件一样被进行读写操作。一次完整的 TCP 套接字操作应该包括创建（socket）、绑定 IP 和端口（bind）、监听（listen）、连

接（connect）、接受（accept）、读写（read、write）、关闭（close）。下面介绍相关结构体和操作 API。

2.1 IPv4 套接字地址结构体 sockaddr_in

结构体原型：

```
struct sockaddr_in {
uint8_t    sin_len;                 /* length of structure */
sa_family_t    sin_family;          /* type of IP, here is AF_INET in IPv4 */
in_port_t    sin_port;              /* port number */
struct in_addr    sin_addr;         /* IP address structure*/
char    sin_zero[8];               /* reserver */
};
struct in_addr
{
 In_addr_t    s_addr;              /* IP address,32-bit */
};
```

这个结构用于保存传输的类型、地址、端口等信息，在多个 API 中需要用到。

2.2 socket( )创建套接字

函数原型：

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

该函数创建一个套接字，成功则返回一个非负整数套接字描述符，失败返回-1。第一个参数为 IP 协议族，与 sockaddr_in 中的 sin_family 保持一致，与 IPv4 对应的参数是 AF_INET；第二个参数一般有两种：SOCK_STREAM 对应 TCP 协议，SOCK_DGRAM 对应 UDP 协议；第三个参数为套接字的属性类型，一般可以设置为 0。

2.3 bind( )绑定端口

函数原型：

```
int bind( int sockfd, const struct sockaddr *myaddr, socklen_t addrlen);
```

该函数将一个套接字绑定到一个 IP 地址和端口号，成功则返回 0，失败返回-1。第一个参数是套接字描述符的值；第二个参数是指向一个套接字地址结构的指针；第三个参数为该套接字地址结构的长度。

对于服务端，通常要指定绑定的端口，可以让内核自动绑定 IP 地址；而对于客户端，可以不指定绑定的 IP 和端口，内核会在 connect 是自动为套接字绑定 IP 和端口。

2.4 connect( )连接服务端

函数原型：

int connect(int sockfd, const struct sockaddr *servaddr, socklen_t addrlen);

该函数用于完成客户端连接服务端的“三次握手”，连接成功返回 0，失败返回-1。第一个参数为客户端套接字描述符的值；第二个和第三个参数为希望连接的服务端 IP 和端口号。

2.5 listen( )服务端监听

函数原型：

int listen(int sockfd, int backlog);

该函数用于使一个套接字从主动状态即客户端状态，转变为被动状态即服务端监听状态，处于监听状态的套接字可以接受其他的套接字的连接请求。监听成功返回 0，出错返回-1。第一个参数为套接字描述符的值，第二个参数为监听的最大可连接数量。

2.6 accept( )服务端接受

函数原型：

int accept(int sockfd, struct sockaddr *cliaddr, socklen_t *addrlen);

该函数用于接受下一个已完成连接的客户端，将客户端的地址信息保存在地址结构中，并返回一个新的套接字描述符，这个新的套接字就是与客户端建立连接的套接字。如果当前没有已完成的连接，函数将会阻塞进程，直到发现已完成连接的客户端。第一个参数为服务端套接字描述符；第二个参数指向的结构体用于保存客

户端地址；第三个参数指向客户端地址长度。其中第二、三参数若为 0 或空指针，则表示服务端不保存客户端地址。

2.7 其他函数

2.7.1 htons( ) / htonl( )

u_short htons(u_short hostshort);

u_long htons(u_long hostlong);

htons( )将主机一个无符号短整型数转化成网络字节顺序，也就是将该数由小端模式转换为大端模式;

2.7.2 inet_pton

int inet_pton(int af, const char *src, void *dst);

该函数将一个字符串形式的“点分十进制”IP 转化为整数形式。第一个参数为地址族；第二个参数指向需要转化的 IP；第三个参数指向转化的整数结果。

2.7.3 select( )

函数原型：

int select(int maxfdpl, fd_set *readset, fd_set *writeset, fd_set *exceptset, const struct timeval *timeout);

select( )函数用来同时等待多个事件的发生，只要其中有一个或多个事件发生，则返回事件数目，若等待时间用完，则返回 0，出错返回-1。

函数的第二、三、四个参数都分别指向一个描述符集。描述符集是一个整数数组，用 fd_set 数据类型表示，数组中的每一个元素的每一位(bit)都对应一个描述符。*readset 中的描述符被指定为可读返回，即当有描述符转变为可读就绪状态时，select 即返回就绪描述符的数目，同理*writeset 中的描述符被指定为可写返回，*exceptset 中的描述符被指定为异常返回。在调用 select 函数之前，要将需要监控的描述符对应的位置 1。

第一个参数需要设置为最大描述符的值加 1，表示从描述符 0 到最大描述符之间的所有描述符都将被监控，而只有对应位设置为 1 的描述符才能使 select 函数返回。

第三个参数为等待时间，通过结构 timeval 进行设置。

```
struct timeval
{
  long tv_sec;                /* seconds */
  long tv_usec;               /* microseconds */
};
```

*timeout 若设置为空指针，则表示函数一直等待，直到事件发生。

为了对描述符集中相应位的置 0 与置 1，以及对相应位的测试，select 提供了四个宏：

1、void FD_ZERO(fd_set &fdset)；

FD_ZERO 用于将制定的描述符集中的所有位清零；

2、void FD_SET(int fd, &fdset);

FD_SET 用于对一个描述符在描述符集中的相应位置 1，即将该描述符加入等待范围；

3、void FD_CLR(int fd, &fdset);

FD_CLR 的作用与 FD_SET 相反，关闭一个描述符的对应位；

4、void FD_ISSET(int fd, &fdset);

FD_ISSET 用于 select 函数返回之后，测试描述符的对应位是否依旧为 1，返回 1 则表示该描述符为就绪状态，返回 0 表示该描述符未就绪。

由于 select 每次返回都把未就绪的描述符位清零，因此每次在调用 select 之前，都应该将需要加入等待的位重新置 1。

三、构建代码

服务端：

1、创建一个套接字。调用 socket( )，根据地址族和传输类型选择相应的参数；

2、绑定 IP 和端口。先设置套接字地址结构 sockaddr 中的信息，再调用 bind( ) 对套接字进行绑定；

3、监听套接字。调用 listen( )将套接字转为监听模式，并设置最大监听数量；

4、等待接受连接。调用 accept( )，阻塞当前进程，直到发现已完成的连接时返回；

5、应用操作。accept( )成功将返回一个新的描述符，对该描述符读写操作即可完成与客户端的通信；

6、应用结束，关闭套接字。

客户端：

1、创建一个套接字。同服务端；

2、连接服务端。在套接字地址结构中添加连接的服务端 IP 和端口，然后调用 connect( )完成连接；

3、应用操作。连接成功即可进行通信；

4、应用结束，关闭套接字。