Socket：

struct sockaddr\_in {

short sin\_family; // 2 bytes e.g. AF\_INET, AF\_INET6

unsigned short sin\_port; // 2 bytes e.g. htons(3490)

struct in\_addr sin\_addr; // 4 bytes see struct in\_addr, below

char sin\_zero[8]; // 8 bytes zero this if you want to

};

**PF\_INET与AF\_INET的区别**：http://blog.csdn.net/jin13277480598/article/details/53842378

理论上建立socket时是指定协议，应该用PF\_xxxx，设置地址时应该用AF\_xxxx。当然AF\_INET和PF\_INET的值是相同的，混用也不会有太大的问题。也就是说你socket时候用PF\_xxxx，设置的时候用AF\_xxxx也是没关系的，这点随便找个TCPIP例子就可以验证出来了。如下，不论是AF\_INET还是PF\_INET都是可行的，只不过这样子的话，有点不符合规范。

**Socket通信中AF\_INET 和 AF\_UNIX域的区别**：https://www.cnblogs.com/langzou/p/7497848.html

1 建立socket传递的地址域，及bind()的地址结构稍有区别：

　　socket() 分别传递不同的域AF\_INET和AF\_UNIX

　　bind()的地址结构分别为sockaddr\_in（制定IP端口）和sockaddr\_un（指定路径名）

2 AF\_INET需经过多个协议层的编解码，消耗系统cpu，并且数据传输需要经过网卡，受到网卡带宽的限制。AF\_UNIX数据到达内核缓冲区后，由内核根据指定路径名找到接收方socket对应的内核缓冲区，直接将数据拷贝过去，不经过协议层编解码，节省系统cpu，并且不经过网卡，因此不受网卡带宽的限制。

3 AF\_UNIX的传输速率远远大于AF\_INET

3 AF\_INET不仅可以用作本机的跨进程通信，同样的可以用于不同机器之间的通信，其就是为了在不同机器之间进行网络互联传递数据而生。而AF\_UNIX则只能用于本机内进程之间的通信。

AF\_UNIX由于其对系统cpu的较少消耗，不受限于网卡带宽，及高效的传递速率，本机通信则首选AF\_UNIX域。不用多说，AF\_INET则用于跨机器之间的通信。

**Inet\_pton点分十进制转换**

inet\_pton是一个IP地址转换函数，可以在将IP地址在“点分十进制”和“二进制整数”之间转换而且，inet\_pton和inet\_ntop这2个函数能够处理ipv4和ipv6。算是比较新的函数了。

inet\_pton(AF\_INET , connect\_ip , &servaddr.sin\_addr)

send函数

int send( SOCKET s,const char\* buf,int len,int flags);

不论是客户还是服务器应用程序都用send函数来向TCP连接的另一端发送数据。

客户程序一般用send函数向服务器发送请求，而服务器则通常用send函数来向客户程序发送应答。

该函数的第一个参数指定发送端套接字描述符；

第二个参数指明一个存放应用程序要发送数据的缓冲区；

第三个参数指明实际要发送的数据的字节数；

第四个参数一般置0。

同步/异步主要针对C端: http://blog.csdn.net/hguisu/article/details/7453390

阻塞/非阻塞主要针对S端:

**异步的概念和同步相对**。当c端一个异步过程调用发出后，调用者不能立刻得到结果。实际处理这个调用的部件在完成后，通过状态、通知和回调来通知调用者。

**阻塞调用**是指调用结果返回之前，当前线程会被挂起（线程进入非可执行状态，在这个状态下，cpu不会给线程分配时间片，即线程暂停运行）。函数只有在得到结果之后才会返回。

有人也许会把**阻塞调用和同步调用**等同起来，实际上他是不同的。对于同步调用来说，很多时候当前线程还是激活的，只是从逻辑上当前函数没有返回而已。 例如，我们在socket中调用recv函数，如果缓冲区中没有数据，这个函数就会一直等待，直到有数据才返回。

非阻塞和阻塞的概念相对应，指在不能立刻得到结果之前，该函数不会阻塞当前线程，而会立刻返回。

1. 同步，就是我客户端（c端调用者）调用一个功能，该功能没有结束前，我（c端调用者）死等结果。

2. 异步，就是我（c端调用者）调用一个功能，不需要知道该功能结果，该功能有结果后通知我（c端调用者）即回调通知。

**文件描述符：**http://blog.csdn.net/wypblog/article/details/6826286

struct fd\_set可以理解为一个集合，这个集合中存放的是文件描述符(file descriptor)，即文件句柄，这可以是我们所说的普通意义的文件，当然Unix下任何设备、管道、FIFO等都是文件形式，全部包括在内，所以毫无疑问一个socket就是一个文件，socket句柄就是一个文件描述符。fd\_set集合可以通过一些宏由人为来操作，比如清空集合 FD\_ZERO(fd\_set \*)，将一个给定的文件描述符加入集合之中FD\_SET(int ,fd\_set \*)，将一个给定的文件描述符从集合中删除FD\_CLR(int ,fd\_set\*)，检查集合中指定的文件描述符是否可以读写FD\_ISSET(int ,fd\_set\* )。