<https://blog.csdn.net/stanjiang2010/article/details/5709882>

 listen系统可以使一台主机上的一个tcp socket在某个端口号被动侦听，等待来自其它主机的tcp socket的连接请求，下面是listen系统调用的函数原型：

#include int listen(int s, int backlog);

   1）backlog是侦听队列的长度，在内核函数中，首先对backlog作检查，如果大于128，则强制使其等于128。

   2）接下来要检查结构体struct sock的成员sk\_state，即当前socket的状态，如果不为TCP\_LISTEN，则开始启动端口侦听。

     启动端口侦听首先要为结构体struct inet\_connection\_sock(它是struc sock的扩展，表示一个面向连接的socket)的成员icsk\_accept\_queue分配内存，icsk\_accept\_queue的类型是struct request\_sock\_queue，

定义如下：

struct request\_sock\_queue

 {

     struct request\_sock \*rskq\_accept\_head;

     struct request\_sock \*rskq\_accept\_tail;

     rwlock\_t syn\_wait\_lock;

     u8 rskq\_defer\_accept;

     struct listen\_sock \*listen\_opt;

};

     tcp socket在侦听的时候，那些来自其它主机的tcp socket的连接请求一旦被接受(完成三次握手协议)，便会建立一个request\_sock，建立与请求socket之间的一个tcp连接。该request\_sock会被放在一个先进先出的队列中，等待accept系统调用的处理。但上面的结构体中好像并没有可以存放request\_sock的地方，

下面是结构体struct listen\_sock的定义：

struct listen\_sock

{

     u8 max\_qlen\_log;

     int qlen;

     int qlen\_young;

     int clock\_hand;

     u32 hash\_rnd;

     u32 nr\_table\_entries;

     struct request\_sock \*syn\_table[0];

 };

     新建立的request\_sock就存放在syn\_table中。这是一个哈希数组，总共有nr\_table\_entries项。实际上在分配内存时，分配的大小是TCP\_SYNQ\_HSIZE（512）项。成员nr\_table\_entries的值是512。成员max\_qlen\_log以2的对数的形式表示request\_sock队列的最大值。哈希表有512项，但队列的最大值的取值是1024。即max\_qlen\_log的值为10。qlen是队列的当前长度。hash\_rnd是一个随机数，计算哈希值用，结构体struct request\_sock\_queue中的rskq\_accept\_head和rskq\_accept\_tail分别指向request\_sock队列的队列头和队列尾。 为struct inet\_connection\_sock分配完内存后，继续处理，结构体struct sock有两个成员sk\_ack\_backlog和sk\_max\_ack\_backlog。sk\_ack\_backlog表示该侦听socket上，当前连向该socket，但是还没有完成三次握手协议的socket的数量，即还在连接过程中的socket的数量。初始值为0，sk\_max\_ack\_backlog为该数量的最大值，也就是listen系统调用的第二个参数，即侦听队列的长度，它的真正含义是：侦听socket能处理的最大并发连接请求数，其最大取值为128。

    到这里，把socket的状态改为TCP\_LISTEN，进入侦听状态。然后独占端口，使socket进入mytcp\_hashinfo哈希表集中的listening\_hash表。侦听建立完成。 由于侦听socket始终在系统中进行侦听工作，所以在进程结束时，还必须显式结束侦听，进行相应的清理工作。