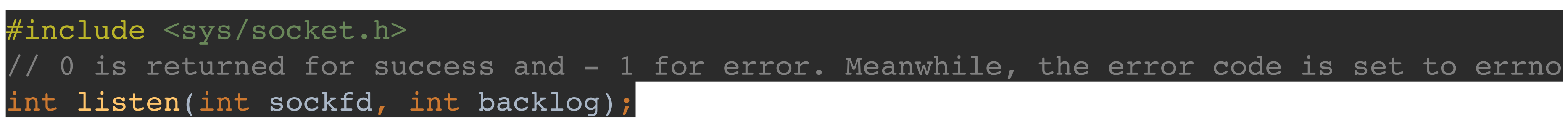
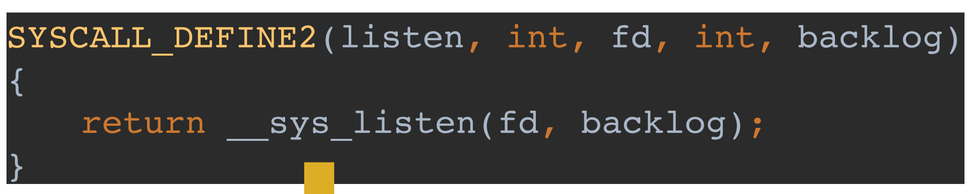
Linux socket listen code:



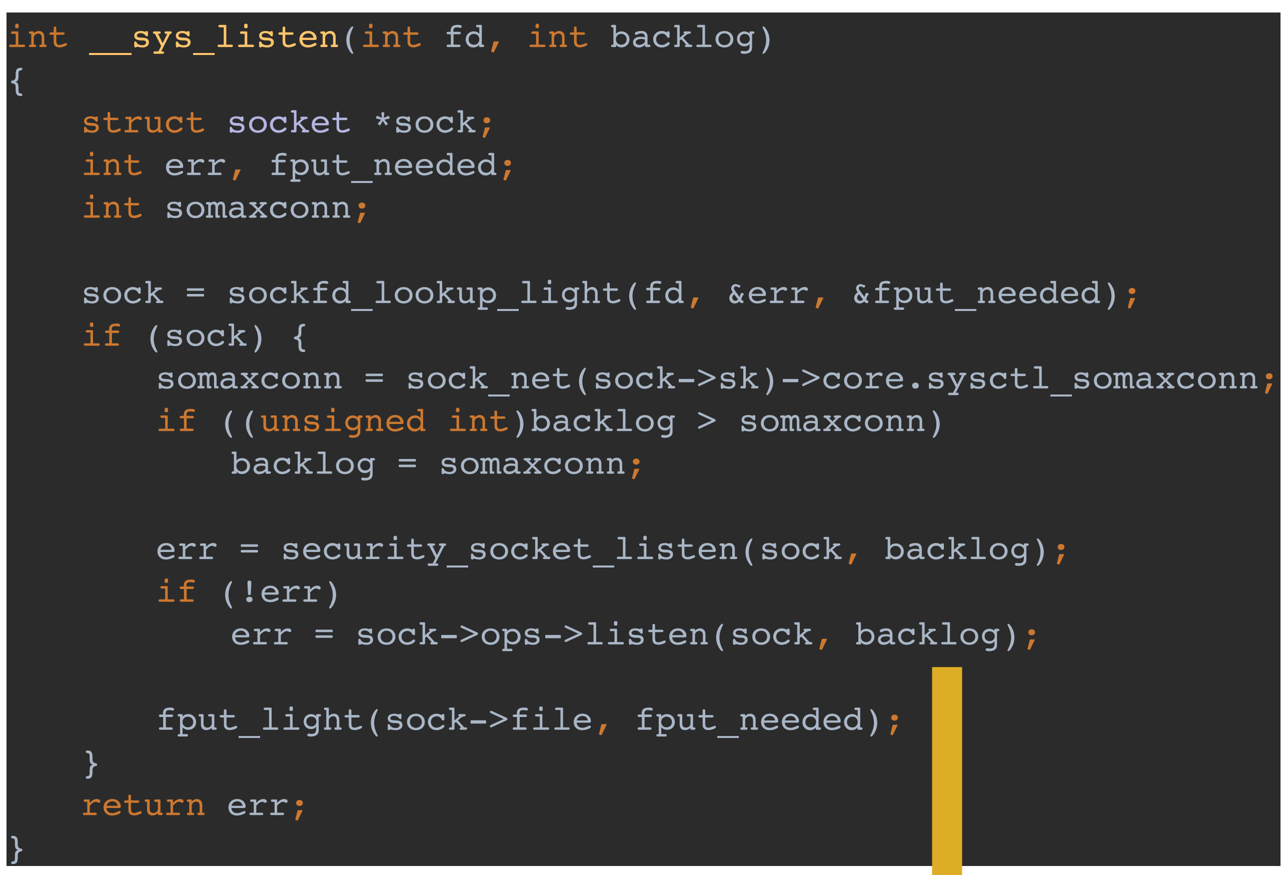
```
#include <sys/socket.h>
// 0 is returned for success and - 1 for error. Meanwhile, the error code is set to errno
int listen(int sockfd, int backlog);
```

调用listen（c语言: libc's INLINE_SYSCALL)
重要参数: backlog (建立连接队列，注意配置)
golang直接读: /proc/sys/net/core/somaxconn



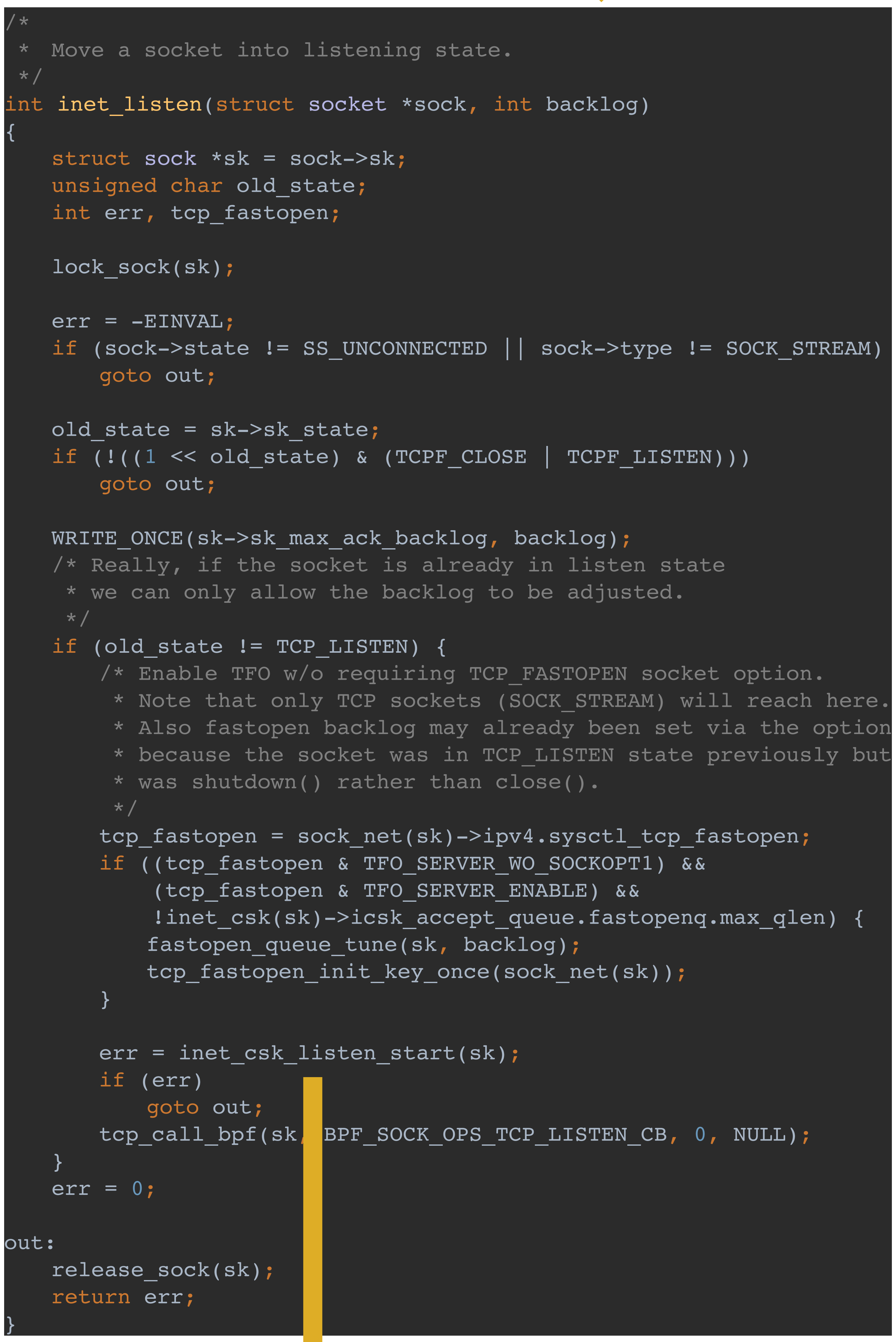
```
SYSCALL_DEFINE2(listen, int, fd, int, backlog)
{
    return __sys_listen(fd, backlog);
}
```

内核代码开始: net/socket.c
定义系统调用



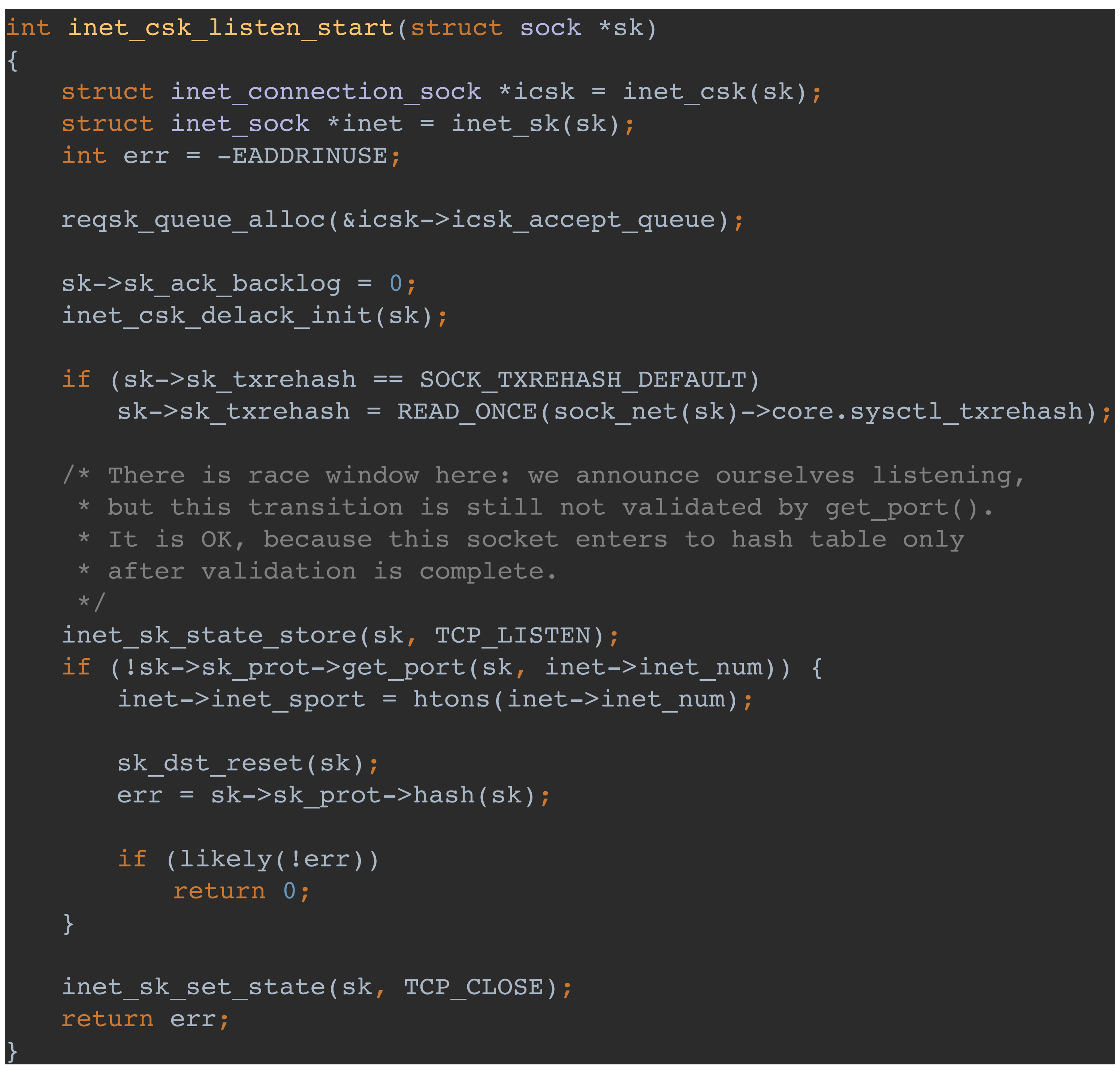
```
int __sys_listen(int fd, int backlog)
{
    struct socket *sock;
    int err, fput_needed;
    int somaxconn;

    sock = sockfd_lookup_light(fd, &err, &fput_needed);
    if (sock) {
        somaxconn = sock_net(sock->sk)->core.sysctl_somaxconn;
        if ((unsigned int)backlog > somaxconn)
            backlog = somaxconn;

        err = security_socket_listen(sock, backlog);
        if (!err)
            err = sock->ops->listen(sock, backlog);

        fput_light(sock->file, fput_needed);
    }
    return err;
}
```

```
listen
    |->INLINE_SYSCALL(listen......)
        |->SYSCALL_DEFINE2(listen, int, fd, int, backlog)
            /* 检查对应的fd是否存在，不存在返回BADF
            |->sockfd_lookup_light
            /* backlog 取 backlog和somaxconn最小值 /proc/sys/net/core/somaxconn
            |->if ((unsigned int)backlog > somaxconn) backlog = somaxconn
            |->sock->ops->listen(sock, backlog) <=> inet_listen
```



```
/*
 *  Move a socket into listening state.
 */
int inet_listen(struct socket *sock, int backlog)
{
    struct sock *sk = sock->sk;
    unsigned char old_state;
    int err, tcp_fastopen;

    lock_sock(sk);

    err = -EINVAL;
    if (sock->state != SS_UNCONNECTED || sock->type != SOCK_STREAM)
        goto out;

    old_state = sk->sk_state;
    if (!((1 << old_state) & (TCPF_CLOSE | TCPF_LISTEN)))
        goto out;

    WRITE_ONCE(sk->sk_max_ack_backlog, backlog);
    /* Really, if the socket is already in listen state
     * we can only allow the backlog to be adjusted.
     */
    if (old_state != TCP_LISTEN) {
        /* Enable TFO w/o requiring TCP_FASTOPEN socket option.
         * Note that only TCP sockets (SOCK_STREAM) will reach here.
         * Also fastopen backlog may already been set via the option
         * because the socket was in TCP_LISTEN state previously but
         * was shutdown() rather than close().
         */
        tcp_fastopen = sock_net(sk)->ipv4.sysctl_tcp_fastopen;
        if ((tcp_fastopen & TFO_SERVER_WO_SOCKOPT1) &&
            (tcp_fastopen & TFO_SERVER_ENABLE) &&
            !inet_csk(sk)->icsk_accept_queue.fastopenq.max_qlen) {
            fastopen_queue_tune(sk, backlog);
            tcp_fastopen_init_key_once(sock_net(sk));
        }

        err = inet_csk_listen_start(sk);
        if (err)
            goto out;
        tcp_call_bpf(sk, BPF_SOCK_OPS_TCP_LISTEN_CB, 0, NULL);
    }
    err = 0;

out:
    release_sock(sk);
    return err;
}
```

net/af_inet.c
listen可以被重复调用
重复调用时，backlog queue可以被修改



```
int inet_csk_listen_start(struct sock *sk)
{
    struct inet_connection_sock *icsk = inet_csk(sk);
    struct inet_sock *inet = inet_sk(sk);
    int err = -EADDRINUSE;

    reqsk_queue_alloc(&icsk->icsk_accept_queue);

    sk->sk_ack_backlog = 0;
    inet_csk_delack_init(sk);

    if (sk->sk_txrehash == SOCK_TXREHASH_DEFAULT)
        sk->sk_txrehash = READ_ONCE(sock_net(sk)->core.sysctl_txrehash);

    /* There is race window here: we announce ourselves listening,
     * but this transition is still not validated by get_port().
     * It is OK, because this socket enters to hash table only
     * after validation is complete.
     */
    inet_sk_state_store(sk, TCP_LISTEN);
    if (!sk->sk_prot->get_port(sk, inet->inet_num)) {
        inet->inet_sport = htons(inet->inet_num);

        sk_dst_reset(sk);
        err = sk->sk_prot->hash(sk);

        if (likely(!err))
            return 0;
    }

    inet_sk_set_state(sk, TCP_CLOSE);
    return err;
}
```

net/inet_connection_sock.c
err = sk->sk_prot->hash(sk):
绑定当前socket到全局hash表
用于收到SYNC保文时寻找