wjlkoorey的博客

wjlkoorey.blog.chinaunix.net

wanderlust in the sea...

wjlkoorey258

原创 揭开网络编程常见API的面纱【上】 2012-08-01 22:09:15

分类： LINUX

Linux网络编程API函数初步剖析

今天我们来分析一下前几篇博文中提到的网络编程中几个核心的API，探究一下当我们调用每个API时，内核中具体做了哪些准备和初始化工作。

博客访问： 509967
博文数量： 97
博客积分： 671
博客等级： 上尉
技术积分： 10228
用 户 组： 普通用户
注册时间： 2010-12-18 16:08

个人简介
www.5678520.com

文章分类
全部博文（97）
Netfilter&ebtabl（0）
算法设计（8）

1、socket(family,type,protocol)

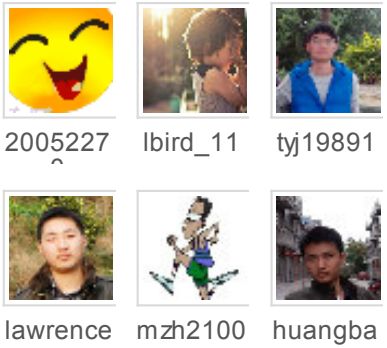
当我们在开发网络应用程序时，使用该系统调用来创建一个套接字。该API所做的工作如下所示：

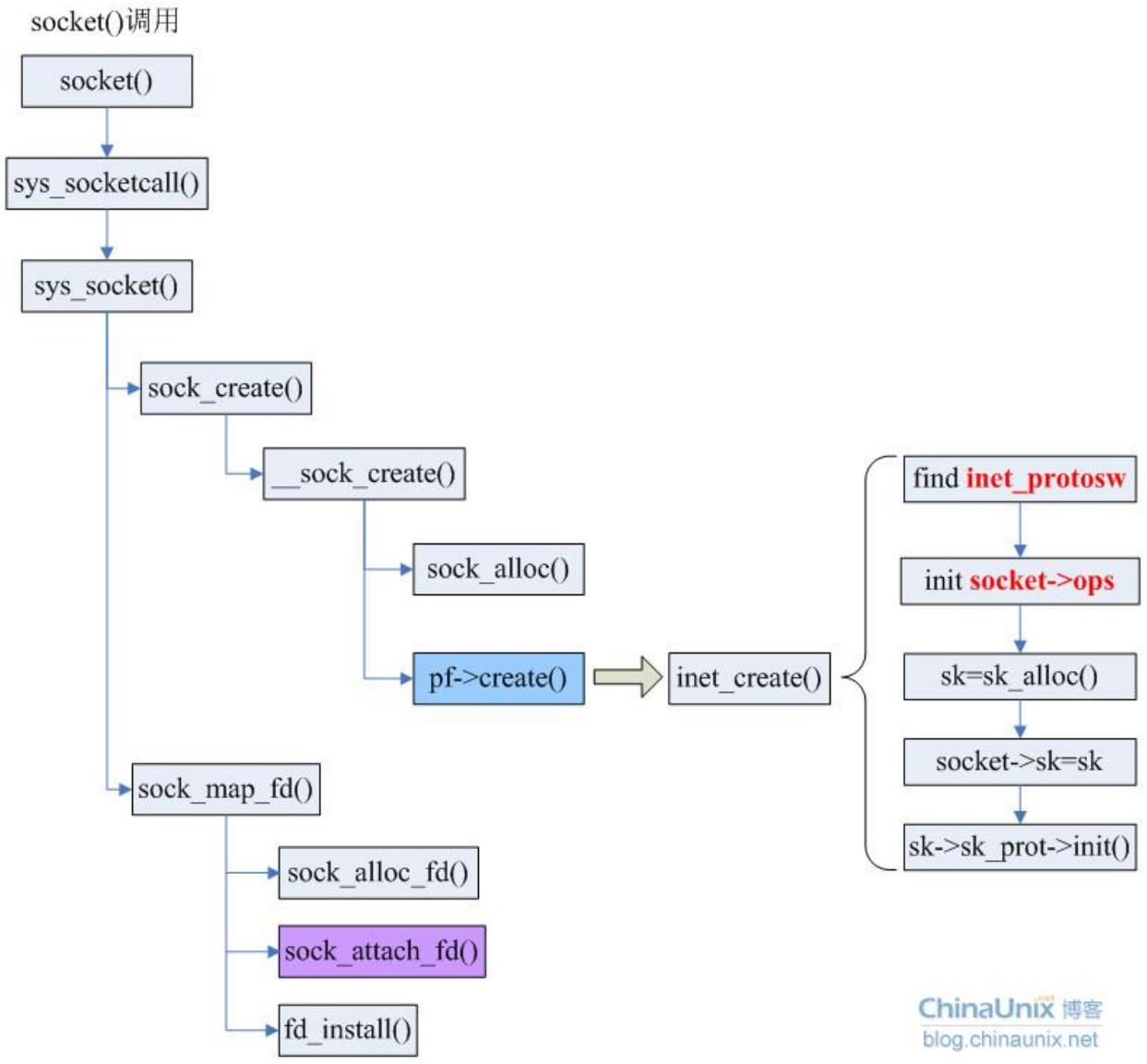
计算机系统（11）
商海ABC（1）
存储（6）
翻译（3）
Java（1）
内核源码（5）
其他（3）
多媒体（7）
网络编程（8）
系统管理（3）
SNMP（2）
Netfilter和iptab（0）
未分配的博文（39）

文章存档

2014年（12）
2013年（21）
2012年（64）

我的朋友





该系统调用主要完成两个任务："创建套接字"和"为套接字绑定文件句柄"。

socket{}<include/linux/net.h>结构定义如下：

```
struct socket {
    socket_state        state;  //socket状态
    unsigned long       flags; //标识：如SOCK_ASYNC_NOSPACE
```

```
    unsigned long           flags; //标识，如SOCK_ASYNC_NOSPACE
    const struct proto_ops  *ops; //协议特定的socket操作集
    struct fasync_struct    *fasync_list; //异步唤醒队列
    struct file             *file; //指向文件的指针
    struct sock             *sk; //指向下一层中的sock结构
    wait_queue_head_t       wait; //等待在这个socket上的任务列表
    short                   type; //数据包的类型
};
```

在创建socket套接字时，就是要完成ops、file和sk等这些成员的初始化。

1). 创建套接字：sock_create()

根据family参数值在全局数组struct net_proto_family *net_families*[]里找到我们所指定的地址簇。不同类型的地址簇都有一个struct net_proto_family{}类型的对象，例如我们常见的IPv4的inet_family_ops，IPv6的inet6_family_ops，X25协议的ax25_family_ops等。在内核是初始化时，这些模块会在自己的初始化函数内部调用sock_register()接口将各自的地址簇对象注册到*net_families*[]数组里。

我们分析的焦点集中在IPv4协议簇，即inet_family_ops对象上。重点是inet_create函数，该函数的主要任务就是创建一个socket套接字，并对其中相关结构体成员进行必要的初始化。至于它创建套接字时的依据和原理等到我们讲协议栈时大家就明白了，这里主要是让大家对其流程执行流程有个感性的把握。

sock_alloc()函数中我们创建一个struct socket{}类型的对象，假如叫做A，将socket()系统调用的第二参数type字段赋值给A->type。

在inet_create()函数中，我们根据type的值，在全局数组struct inet_protosw *inetsw*[]里找到我们对应的协议转换开关。而inetsw[]数组是在inet_init()函数里被初始化的：

·LR模型的Spark实现
·对Oracle高水位线的研究实践...
·为学习Hadoop使用VMware准备3...
·【故障处理】opmn启动失败及...
·oracle 11g ASM 磁盘组在线扩...
·数据迁移中的数据库检查和建...

热词专题

·李体育老师荣获“杰出传承人...
·string.h
·安装oracle
·VMware VDP
·Android + 系统属性

```
for (q = inetsw_array; q < &inetsw_array[INETSW_ARRAY_LEN]; ++q)
    inet_register_protosw(q);
```

其中inetsw_array[]是一个比较重要的数据结构，定义在af_inet.c文件中：

```
static struct inet_protosw inetsw_array[] =
{
    {
        .type =       SOCK_STREAM,
        .protocol =   IPPROTO_TCP,
        .prot =       &tcp_prot,
        .ops =        &inet_stream_ops,
        .capability = -1,
        .no_check =   0,
        .flags =      INET_PROTOSW_PERMANENT |
                      INET_PROTOSW_ICSK,
    },

    {
        .type =       SOCK_DGRAM,
        .protocol =   IPPROTO_UDP,
        .prot =       &udp_prot,
        .ops =        &inet_dgram_ops,
        .capability = -1,
        .no_check =   UDP_CSUM_DEFAULT,
        .flags =      INET_PROTOSW_PERMANENT,
    },


    {
        .type =       SOCK_RAW,
        .protocol =   IPPROTO_IP,    /* wild card */
        .prot =       &raw_prot,
        .ops =        &inet_sockraw_ops,
        .capability = CAP_NET_RAW,
        .no_check =   UDP_CSUM_DEFAULT,
        .flags =      INET_PROTOSW_REUSE,
    }
};
```


根据type的值，就可以确定struct socket{}->ops，到底是inet_stream_ops、inet_dgram_ops或者inet_sockraw_ops。然后，对应地，就以tcp_prot、udp_prot或raw_prot为输入参数，实例化一个struct sock{} 对象sk=sk_alloc()。紧接着建立socket{}和sock{}的关联，最后将socket()系统调用的第三个参数protocol付给sock{}对象中的属性sk_protocol。

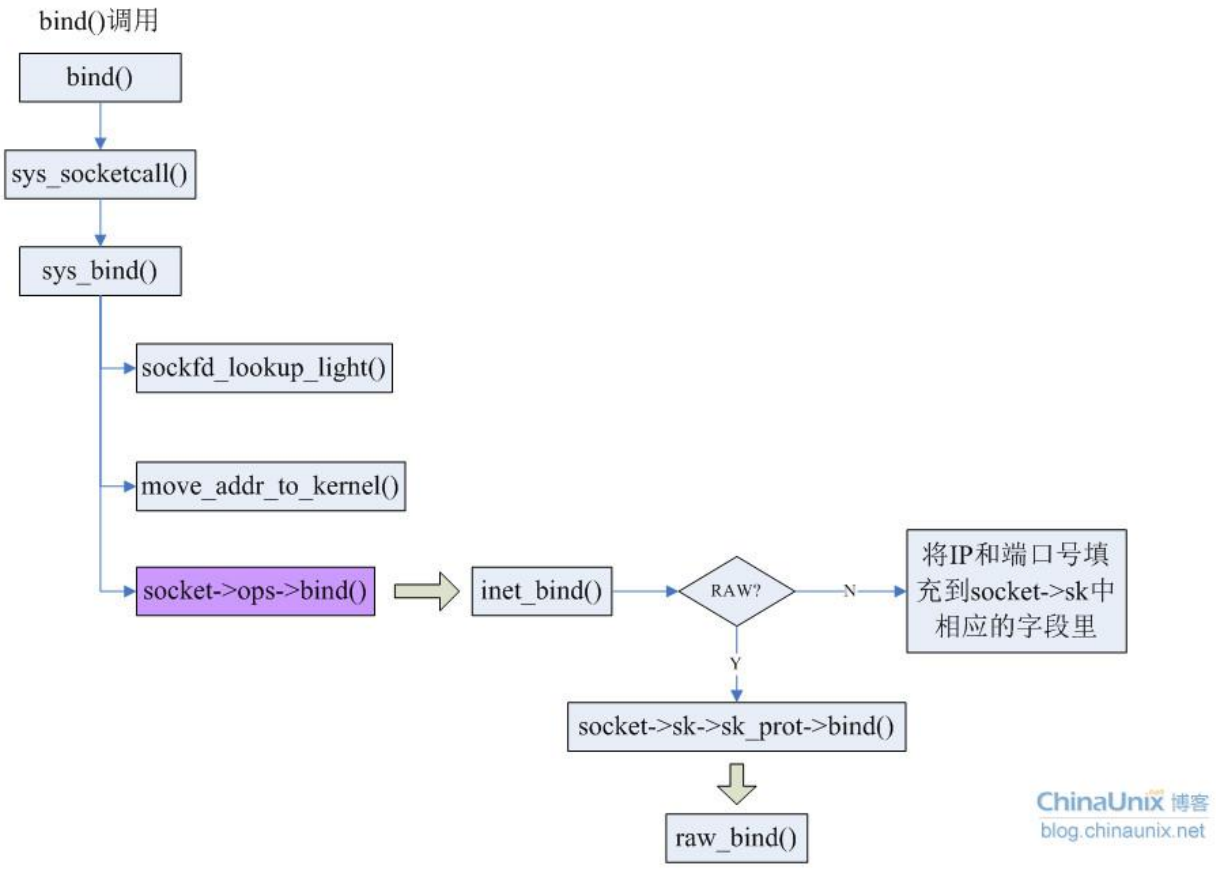
看不懂别着急，我说过，这里只是给大家梳理整体流程，等到我们讲了协议栈章节，然后再回头看本篇，就感觉这些东西就太小儿科了。

2). 为套接字绑定文件句柄：sock_map_fd()

我们都知道网络套接字也是一种系统IO，所以不可避免的要与文件系统打交道。每个套接字都对应一个已打开的文件标识符，所以在套接字初始化完成后，就要将其和本地一个唯一的文件标识符关联起来，即建立socket{}和file{}之间的关联关系。

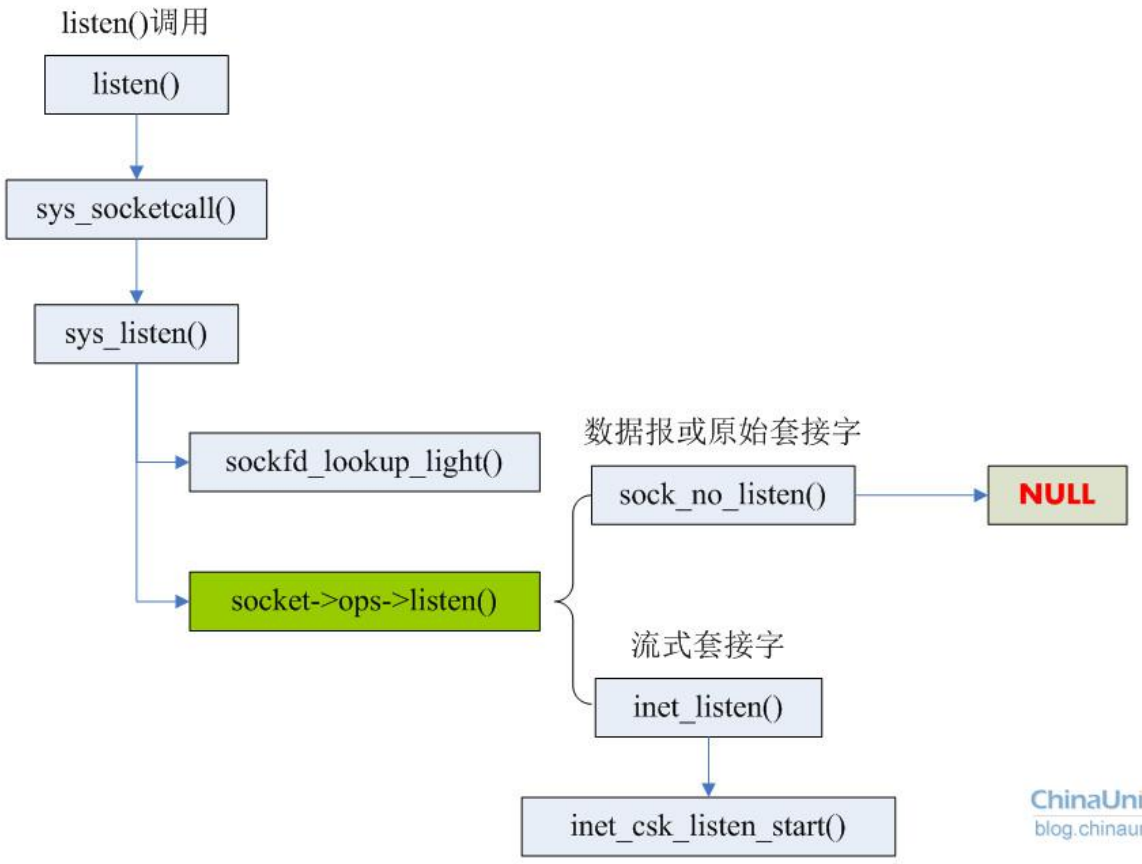
2、bind (sockfd, sockaddr, addrlen)

该系统调用在内核中的执行过程如下：



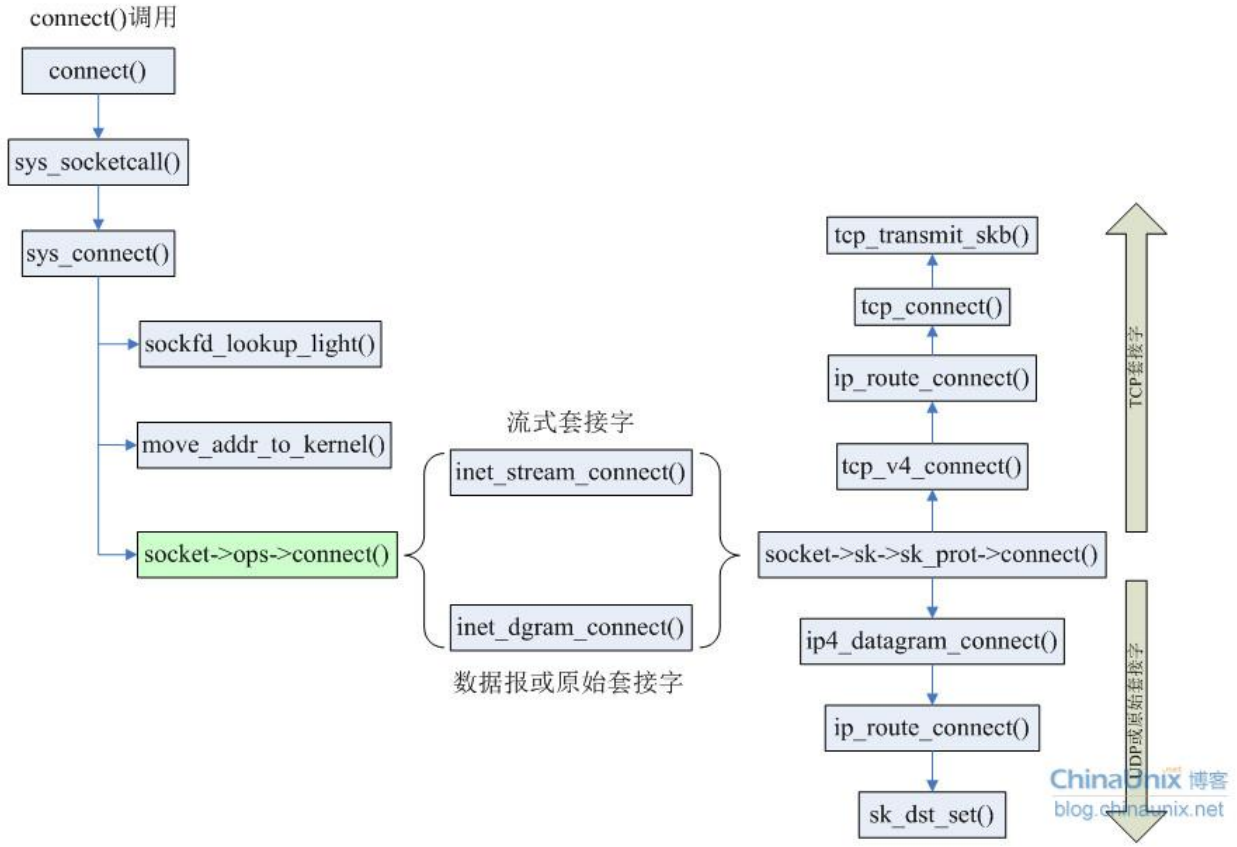
重点是socket->ops->bind()回调接口。我们现在已经知道了，针对IPv4而言，这里的ops无非就是***inet_stream_ops***、***inet_dgram_ops***或***inet_sockraw_ops***对象。碰巧的是，这三个对象中的bind函数指针均指向inet_bind()函数。只有原始套接字的情况，这里会去调用***raw_prot***对象的bind回调函数，即raw_bind()。

3、listen(sockfd, backlog)



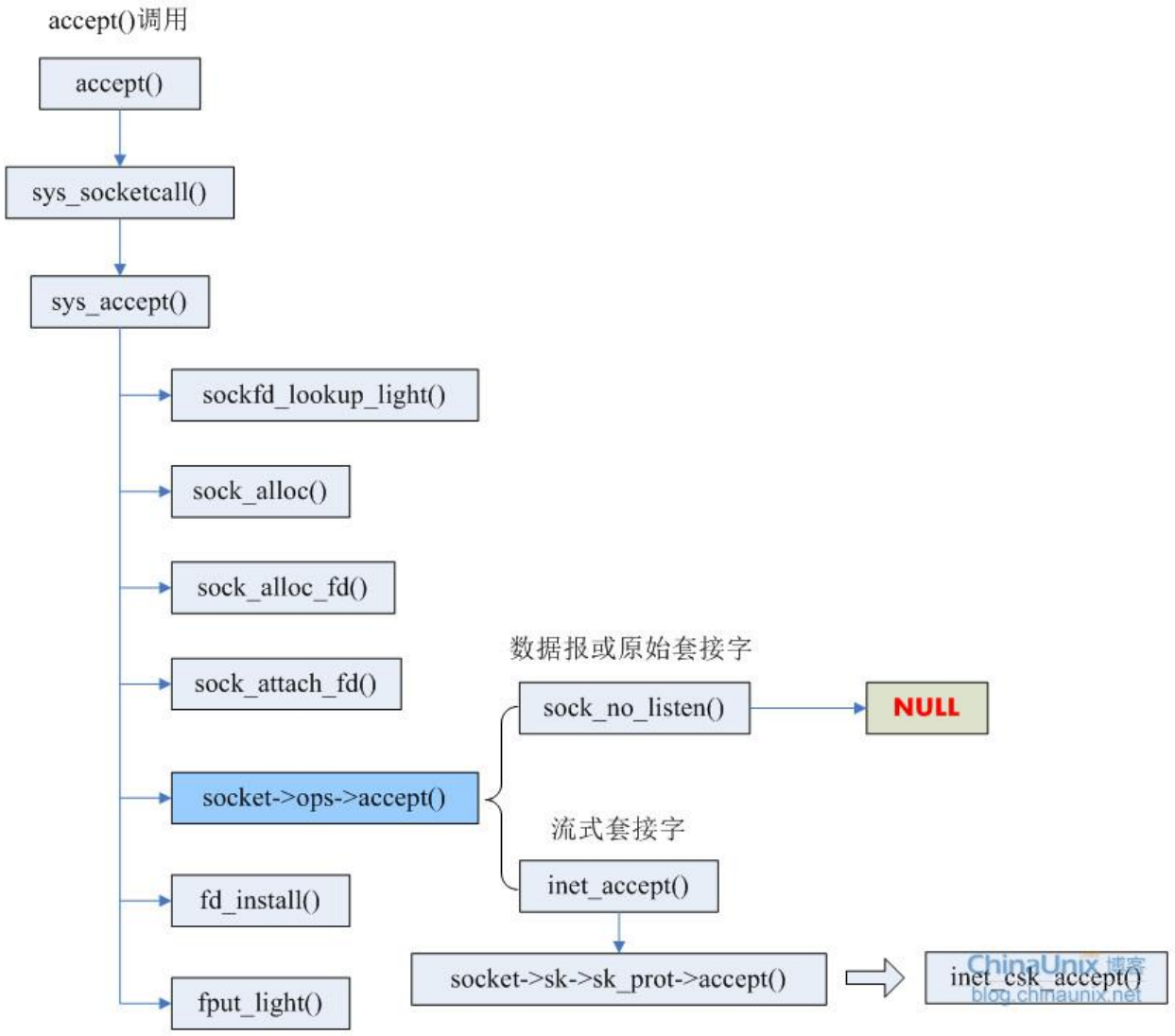
这里我们可以看到面向无连接的套接字和原始套接字是不用listen的，只有流式套接字才有效。

4、connect(sockfd, sockaddr, addrlen)



从这幅图中我们确实看到，connect()系统调用不但可以面向连接的套接字，也可用于无连接及原始套接字。

5、accept(sockfd, sockaddr, addrlen)



同样地，我们看到只有面向连接的流式套接字调用accept()才有意义。最终调用的是***tcp_prot***对象的accept成员函数。

本篇主要进一步分析了网络编程中常见的几个API函数内部的调用流程，一方面可以使大家对这些API的有了更深的认识，不是仅仅停留在形而上的层面；另一方面为后面分析协议栈的实现原理，奠定坚实的基础。

未完，待续...

上一篇：Linux网络编程：原始套接字的魔力【续】
下一篇：揭开网络编程常见API的面纱【下】

相关热门文章

热 轻量级web server Tornado代码...
荐 linux 常见服务端口
热 kernel 报错l701.exel[16922]:...
热 《午夜计程车》网台联播...
荐 【ROOTFS搭建】busybox的httpd...
热 C语言 如何在一个整型左边补0...
热 unix网络编程一卷 unp.h...
荐 xmanager 2.0 for linux配置
热 python无法爬取阿里巴巴的数据...
热 众安在线揭开神秘面纱 首批保...
荐 什么是shell
热 linux-2.6.28 和linux-2.6.32....
热 中国尿素交易网
荐 linux socket的bug??
热 linux su - username -c 命...

给主人留下些什么吧！~~

uahorngt 2014-05-11 14:00:51

分享了，学习中，希望有所帮助，谢谢。。 隆达木业，创造绿色新生活
Longda Wooden manufacture corporation limited </s
pan>www.longda-
wood.com Our factory Longda Wood Co., Ltd is located in Fujian Province
s, where produces poplar and hardwood, and&n

回复 | 举报

评论热议

请登录后评论。

登录 注册