< 2016年7月 >

日	一	二	三	四	五	六
26	27	28	29	30	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31	1	2	3	4	5	6

搜索

找找看

常用链接

我的随笔
我的评论
我的参与
最新评论
我的标签
更多链接

文章分类(50)

Linux 配置与应用(18)
Linux 设备驱动(11)
Linux 网络协议栈(3)
TCP / IP(2)
数据结构与算法(5)
数据库(1)
网络编程(10)

文章档案(50)

2012年8月 (4)
2010年7月 (2)
2010年1月 (8)
2009年12月 (6)
2009年11月 (30)

Linux 网络领域

epoll
Linux 网络驱动
Linux内核路由
linux网络kernel
Linux下ip命令手册

sk_buff结构分析

Posted on 2009-11-01 22:08 放飞自我 阅读(15565) 评论(2) 编辑 收藏

前言：

以下是根据《深入理解Linux网络技术内幕》对sk_buff的相关总结，由于是刚刚看这本书（太厚了），不免在前期出现错误，随着对此书的深入我会在修改前面的错误，也希望各位牛人给予指点。帮助我成长。

sk_buff分析：

sk_buff是Linux网络代码中最重要的结构体之一。它是Linux在其协议栈里传送的结构体，也就是所谓的“包”，在他里面包含了各层协议的头部，比如ethernet, ip ,tcp ,udp等等。也有相关的操作等。熟悉他是进一步了解Linux网络协议栈的基础。

此结构定义在<include/linux/skbuff.h>头文件中，结构体布局大致可分为以下四部分：

- 布局（layout）
- 通用（general）
- 功能专用（feature-specific）
- 管理函数（management functions）

网络选项以及内核结构

我们可以看到在此结构体里有很多预处理，他是在需要指定相应功能时才起作用，我们在这里先对通用的作出分析。

布局字段：

sk_buff是一个复杂的双向链表，在他结构中有next和prev指针，分别指向链表的下一个节点和前一个节点。并且为了某些需求（不知道是哪些目前）需要很快定位到链表头部，所以还有一个指向链表头部的指针list(我在2.6.25内核没有发现这个指针)。

sk_buff_head结构是：

```
struct sk_buff_head {
    /* These two members must be first. */
```

Linux中的通知链技术
netlink
netlink 编程介绍
visualsvn 使用
网络流量统计程序

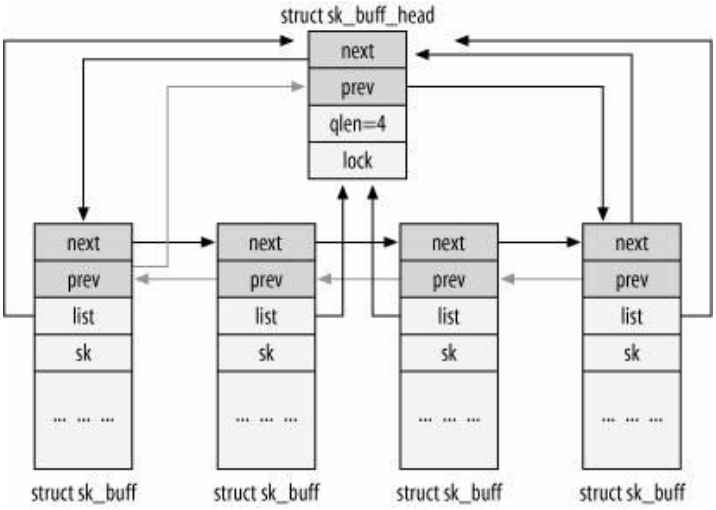
Linux 网络应用
http cache
http cache
http编码
nginx 正向代理服务器配置
nginx源码分析

编程进阶
Little_endian
vim+ctags+cscope
打造vim IDE
字节序

博客收集
linux协议栈函数介绍
vim高级进阶
嵌入式linux开发研究园
网卡驱动博客
网络，mysql，内核等
无名大神的博客

最新评论
1. Re:链表反转
逻辑很清晰，不过我觉得代码还是可以再稍微的优化一下的。if(pNext == NULL) pReversedHead = pNode;链表的头结点指针的这个赋值操作，在while循环中会被多次.....
--xujingreate
2. Re:LINUX netlink机制
@Tsihang很少来这里了，直接下载linux内核代码即可...
--放飞自我

```
    struct sk_buff *next;
    struct sk_buff *prev;
    __u32       qlen; //代表元素节点数目
    spinlock_t      lock; //加锁，防止对表的并发访问
};
```



struct sock *sk

这个指针指向一个套接字sock数据结构。当数据在本地产生或者本地进程接受时，需要这个指针；里面的数据会有tcp/udp和用户态程序使用。如果是转发此指针为NULL

unsigned int len

缓冲区中数据块大小。长度包括：主要缓冲区（head所指）的数据以及一些片断（fragment）的数据。当包在协议栈向上或向下走时，其大小会变，因为有头部的丢弃和添加。

unsigned int data_len

片段中数据大小

unsigned int mac_len

mac包头大小

atomic_t users

引用计数，使用这个sk_buff的使用者的数目，可能有多个函数要使用同一个sk_buff所以防止提前释放掉，设置此计数

unsigned int truesize

此缓冲区总大小，包括sk_buff。sk_buff只不过是个指针的集合，他所指的才是真正的数据区，所以是两部分。（见下图）

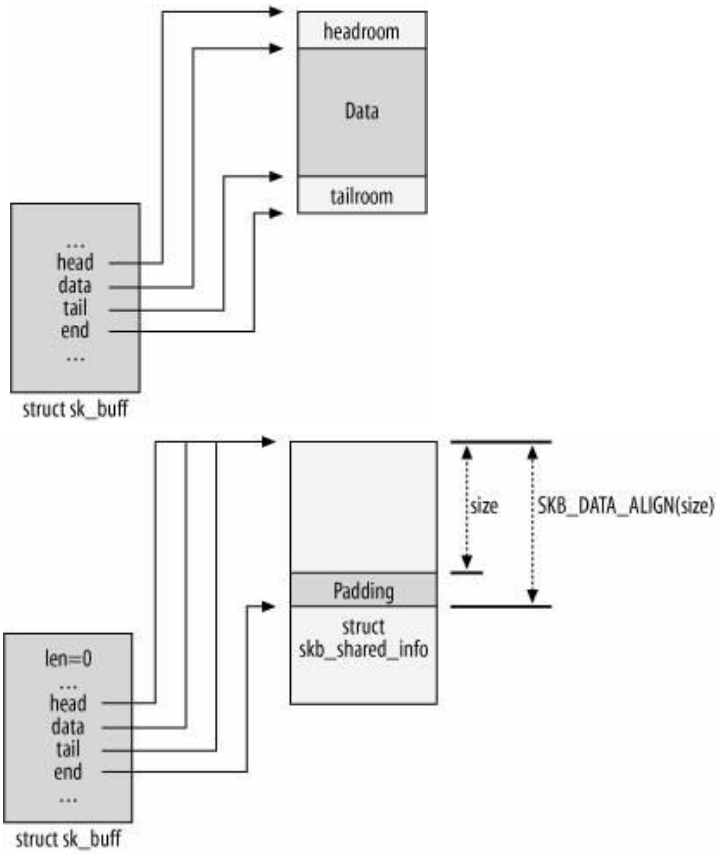
sk_buff_data_t tail;

sk_buff_data_t end;

unsigned char *head, *data;

这些指针很重要，他们指向的是真正的数据区，他们的边界。head和end指向的是数据区的开端和尾端（注意和data，tail区别）如下图，data和tail指向的是实际数据的开头和结尾。

因为数据区在协议栈走的时候要一层层添加或去掉一些数据（比如报头）所以申请一块大的足够的内存，然后在往里放东西。真实的实际数据可能用不了这么多，所以用data,tail指向真实的，head,tail指向边界。刚开始没填充数据时前三个指针指向的是一个地方。



void (*destructor) (…….)

此函数指针被初始化一个函数，当此缓冲区删除时，完成某些工作。

通用字段

struct timeval stamp（2.6.25没有，估计是ktime_t tstamp）

时间戳，表示何时被接受或有时表示包预定的传输时间

struct net_device *dev

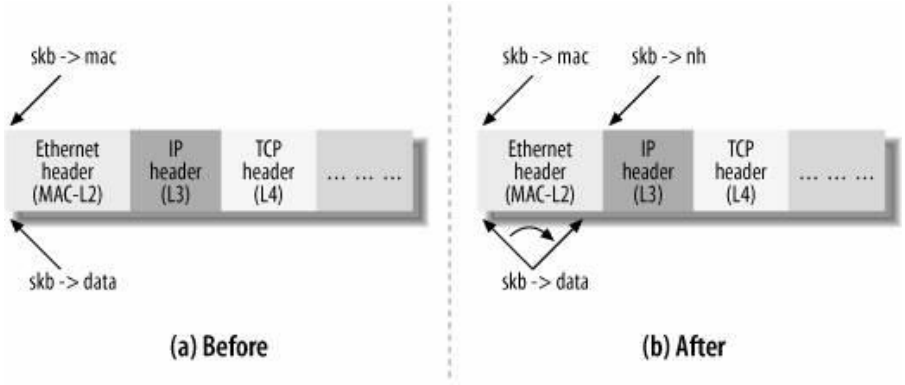
描述一个网络设备，我会以后分析他。

sk_buff_data_t transport_header; //L4

sk_buff_data_t network_header; //L3

sk_buff_data_t mac_header; //L2

这些指针分别指向报文头部，和2.4版本比较有了变化，不再是联合体，使用更加方便了，Linux给出了很方便的函数直接定位到各层的头部。下图是2.4版本的，只是说明一下。



struct dst_entry dst

路由子系统使用。目前不知道怎么回事呢。据说比较复杂。

char cb[40]

缓冲控制区，用来存储私有信息的空间。比如tcp用这个空间存储一个结构体tcp_skb_cb ，可以用宏TCP_SKB_CB(__skb)定位到他，然后使用里面的变量。

ip_summed:2

__wsum csum;

校验和

unsigned char pkt_type

根据L2层帧的目的地址进行类型划分。

unsigned char cloned

表示该结构是另一个sk_buff克隆的。

__u32 priority;

QoS等级

__be16 protocol;

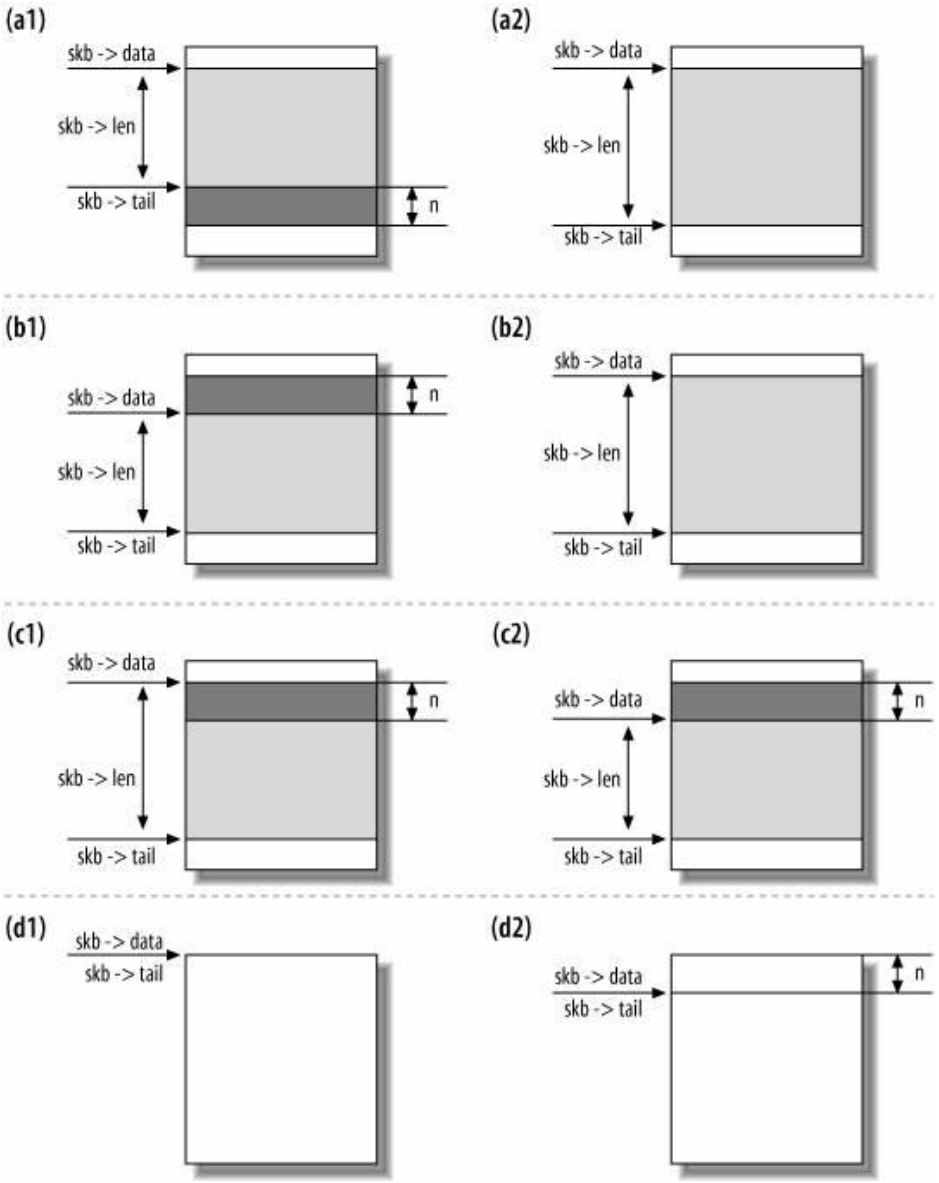
从L2层设备驱动看使用在下一个较高层的协议。

功能专用字段

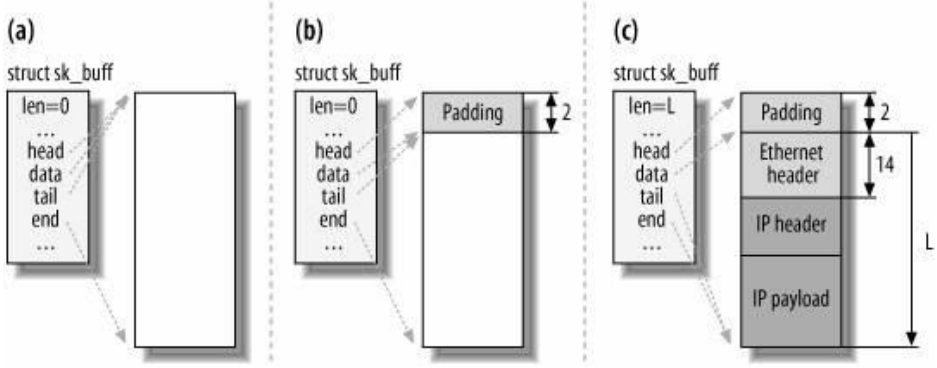
Linux是模块化的，你编译时可以带上特定功能，比如netfilter等，相应的字段才会生效。应该是那些预定义控制的。

管理函数

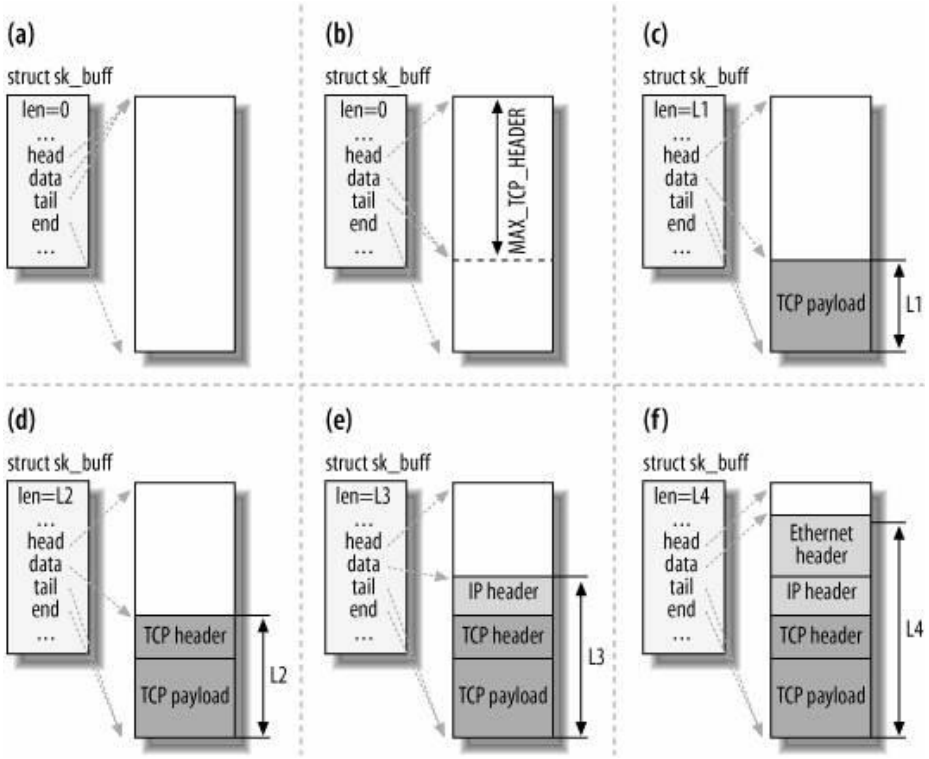
下面这个图是:（a*）skb_put; (b*) skb_push; (c*) skb_pull (d*) skb_reserve的使用，主要是对skb_buf所指向的数据区的指针移动。（数据预留以及对齐）



下图是用skb_reserve函数，把一个14字节的ethernet帧拷贝到缓冲区。skb_reserve(skb, 2)， 2表示16字节对齐。14+2=16



下图是穿过协议栈从tcp层向下到链路层的过程



分配内存：

alloc_skb 分配缓冲区和一个sk_buff结构

dev_alloc_skb 设备驱动程序使用的缓冲区分配函数

释放内存：

kfree_skb 只有skb->users计数器为1时才释放

dev_kfree_skb

缓冲区克隆函数 skb_clone

列表管理函数：

skb_queue_head_init

队列初始化

skb_queue_head , skb_queue_tail

把一个缓冲区添加到队列头或尾

skb_dequeue, skb_dequeue_tail

从头或尾去掉

skb_queue_purge

把队列变空

skb_queue_walk

循环队列每个元素

代码

写完了，O(∩_∩)O~

分类: Linux 网络协议栈

好文要顶 关注我 收藏该文

放飞自我
关注 - 7
粉丝 - 24
+加关注

4 0

(请您对文章做出评价)

Feedback

#1楼[楼主]
2009-11-05 09:51 by 放飞自我

没人顶啊？沙发自己坐吧。

支持(0) 反对(0)

#2楼[楼主]
2010-07-06 16:32 by 放飞自我

目前言论为测试
xxx:http://www.cnblogs.com/iceocean/

支持(0) 反对(0)

刷新评论 刷新页面 返回顶部