skb - Linux network buffers

Harald Welte laforge@gnumonks.org

1.3, 2000/10/14 21:27:02

Short description about the linux network buffers (skb's)

Contents

1 Introduction

At the time I wanted to know more about the Linux network stack, I always wanted a document like this to exist. But unfortunately I never found one. After I gained some basic knowledge about the Linux network stack internals, I wrote one.

I'm happy if this document is of any use for other people trying to learn about the Linux kernel.

Please let me know of any bugs in this document. It should resemble kernel revision 2.4.0-test4

2 skbuff's

skbuffs are the buffers in which the linux kernel handles network packets. The packet is received by the network card, put into a skbuff and then passed to the network stack, which uses the skbuff all the time.

2.1 struct sk_buff

The struct sk_buff is defined in <linux/skbuff.h> as follows:

next

next buffer in list

prev

previous buffer in list

list

list we are on

sk

socket we belong to

stamp

timeval we arrived at

dev

device we are leaving by

rx_dev

device we arrived at

h

transport layer header (tcp,udp,icmp,igmp,spx,raw)

nh

network layer header (ip,ipv6,arp,ipx,raw)

mac

link layer header

dst

FIXME:

cb

control buffer, used internally

len

length of actual data

csum

checksum

used

FIXME: data moved to user and not MSG_PEEK

is_clone

we are a clone

cloned

head may be cloned

pkt_type

packet class

ip_summed

driver fed us ip checksum

priority

packet queuing priority

users

user count

protocol

packet protocol from driver

security

security level of packet

truesize

real size of the buffer

head

pointer to head of buffer

data

data head pointer

tail

tail pointer

end

end pointer

destructor

destructor function

nfmark

netfilter mark

nfcache

netfilter internal caching info

nfct

associated connection, if any

tc_index

traffic control index

2.2 skb support functions

There are a bunch of skb support functions provided by the sk_buff layer. I briefly describe the most important ones in this section.

2.2.1 allocation / free / copy / clone and expansion functions

struct sk_buff *alloc_skb(unsigned int size, int gfp_mask)

This function allocates a new skb. This is provided by the skb layer to initialize some privat data and do memory statistics. The returned buffer has no headroom and a tailroom of /size/ bytes.

void kfree_skb(struct sk_buff *skb)

Decrement the skb's usage count by one and free the skb if no references left.

struct sk_buff *skb_get(struct sk_buff *skb)

Increments the skb's usage count by one and returns a pointer to it.

struct sk_buff *skb_clone(struct sk_buff *skb, int gfp_mask)

This function clones a skb. Both copies share the packet data but have their own struct sk_buff. The new copy is not owned by any socket, reference count is 1.

struct sk_buff *skb_copy(const struct sk_buff *skb, int gfp_mask)

Makes a real copy of the skb, including packet data. This is needed, if You wish to modify the packet data. Reference count of the new skb is 1.

struct skb_copy_expand(const struct sk_buff *skb, int new_headroom, int new_tailroom, int gfp_mask)

Make a copy of the skb, including packet data. Additionally the new skb has a haedroom of /new_headroom/ bytes size and a tailroom of /new_tailroom/ bytes.

2.2.2 anciliary functions

int skb_cloned(struct sk_buff *skb)

Is the skb a clone?

int skb_shared(struct sk_Buff *skb)

Is this skb shared? (is the reference count > 1)?

2.2.3 operations on lists of skb's

struct sk_buff *skb_peek(struct sk_buff_head *list_)

peek a skb from front of the list; does not remove skb from the list

struct sk_buff *skb_peek_tail(struct sk_buff_head *list_)

peek a skb from tail of the list; does not remove sk from the list

__u32 skb_queue_len(sk_buff_head *list_)

return the length of the given skb list

void skb_queue_head(struct sk_buff_head *list_, struct sk_buff *newsk)

enqueue a skb at the head of a given list

void skb_queue_tail(struct sk_buff_head *list_, struct sk_buff *newsk)

enqueue a skb at the end of a given list.

struct sk_buff *skb_dequeue(struct sk_buff_head *list_)

dequeue a skb from the head of the given list.

struct sk_buff *sbk_dequeue_tail(struct sk_buff_head *list_)

dequeue a skb from the tail of the given list

2.2.4 operations on skb data

unsigned char *skb_put(struct sk_buff *sbk, int len)

extends the data area of the skb. if the total size exceeds the size of the skb, the kernel will panic. A pointer to the first byte of new data is returned.

unsigned char *skb_push(struct sk_buff *skb, int len)

extends the data area of the skb. if the total size exceeds the size of the skb, the kernel will panic. A pointer to the first byte of new data is returned.

unsigned char *skb_pull(struct sk_buff *skb, int len)

remove data from the start of a buffer, returning the bytes to headroom. A pointr to the next data in the buffer is returned.

int skb_headroom(struct sk_buff *skb)

return the amount of bytes of free space at the head of skb

int skb_tailroom(struct sk_buff *skb)

return the amount of bytes of free space at the end of skb

struct sk_buff *skb_cow(struct sk_buff *skb, int headroom)

if the buffer passed lacks sufficient headroom or is a clone it is copied and additional headroom made available.