

PB173 – Ovladače jádra – Linux

XII. Síťové rozhraní

Jiri Slaby

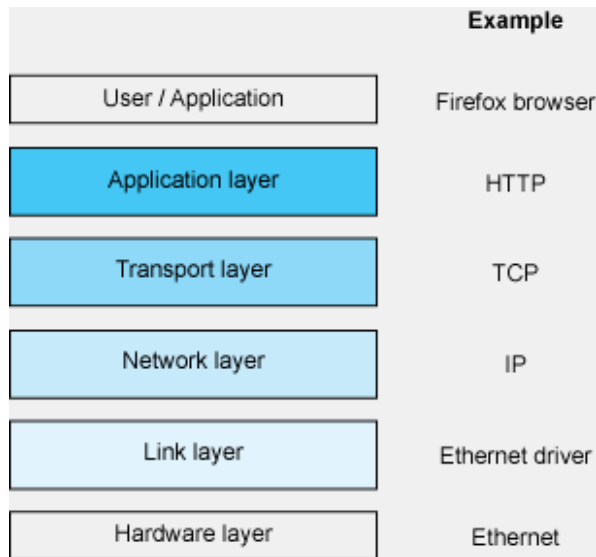
Fakulta informatiky
Masarykova univerzita

9. 12. 2014

LDD3 kap. 17

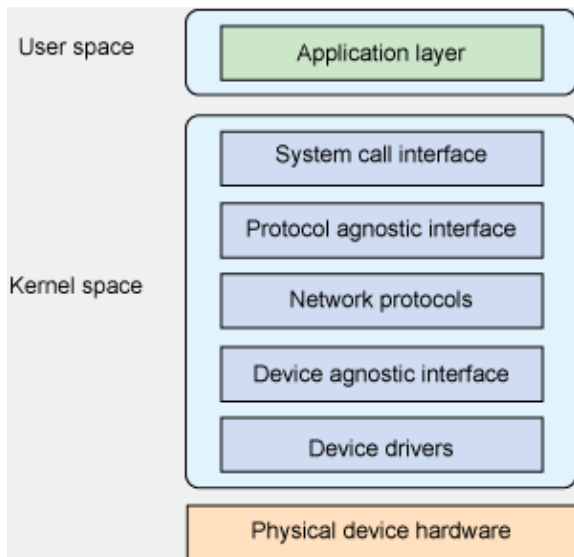
- Ovladač pro ethX

Teoretická síťová vrstva



Zdroj: ldn.linuxfoundation.org

Linuxová síťová vrstva



Zdroj: ldn.linuxfoundation.org

Ovladač síťové karty

- Mezivrstva mezi HW a Linuxem (síťovou vrstvou)
- Vytváří `eth*` a podobná zařízení (tzv. `netdevice`)
- Obsahuje háčky
 - Zařízení je UP, DOWN, změna MTU, pošli paket, ...

API

- `linux/netdevice.h`, `struct net_device`
- Alokace: `alloc_netdev_D`, `free_netdev_D`
- Registrace: `register_netdev`, `unregister_netdev`
- Háčky: `struct net_device_ops` (podobně jako znaková zařízení)

struct net_device_ops

struct net_device_ops

```
int (*ndo_open)(struct net_device *dev); /* ip link set up */
int (*ndo_stop)(struct net_device *dev); /* ip link set down */

/* send a packet */
netdev_tx_t (*ndo_start_xmit)(struct sk_buff *skb, struct net_device *dev);

/* change MTU */
int (*ndo_change_mtu)(struct net_device *dev, int new_mtu);

... /* no receive packet */
```

- Místo `alloc_netdev` se použije konkrétnější `alloc_*`
 - Inicializuje `netdevice`
 - Délku hlavičky, adresy, fronty, ...
- Příklady
 - *Ethernet*: `alloc_etherdev` (`linux/etherdevice.h`)
 - IRDA: `alloc_irdadev` (`net/irda/irda_device.h`)
 - CAN: `alloc_candev` (`linux/can/dev.h`)
 - HDLC: `alloc_hdlcdev` (`linux/hdlc.h`)
 - ...

Postup vytvoření ethernetového zařízení v systému

- 1 Alokace (`alloc_etherdev(priv_size)`)
 - Alokuje navíc paměť o velikosti `priv_size`
 - `priv_size` může být 0
- 2 Inicializace informací
 - Nastavení háčeků (`net_device_ops` do `dev->netdev_ops`)
 - Nastavení MAC adresy (`dev->dev_addr`)
 - Získání ukazatele na paměť alokovanou navíc
 - Funkcí `netdev_priv`
 - Jen při alokaci s `priv_size > 0`
- 3 Registrace (`register_netdev`)
- 4 ...
- 5 Deregistrace (`unregister_netdev`)
- 6 Uvolnění (`free_netdev`)

Vytvoření eth*

- 1 Dle postupu z předchozího slajdu
 - Alokace+registrace v `module_init`
 - Deregistrace+uvolnění v `module_exit`
 - MAC: `random_ether_addr`
 - `priv_size`: `sizeof(struct timer_list)`
 - Nezapomeňte na `setup_timer`
- 2 Vytvořit `open`, `stop`, `start_xmit`
 - Těla vypíší jen název funkce
 - `start_xmit` vrátí `NETDEV_TX_OK`
- 3 NEVKLÁDEJTE do systému
 - Nelze modul odebrat

Tzv. socket buffery

- `linux/skbuff.h`, `struct sk_buff`
- Obecně: `dev_alloc_skbD`
- *Pro netdevice*: `netdev_alloc_skbD`
- Uvolnění: `dev_kfree_skbD`
- Obsah: `skb->data`
 - Ukazuje na alokovanou paměť
 - Plní/čte se klasicky přes `memcpy` apod.
- Délka obsahu: `skb->len`
- Protokol: `skb->protocol`
 - Při příjmu nutno nastavit!

skb **je parametr** `start_xmit`

1 Předat do HW data

- `skb->data` o délce `skb->len`

2 Uvolnit skb

- `dev_kfree_skb(skb)`

3 Vrátit stav

- `NETDEV_TX_OK`
- `NETDEV_TX_BUSY`

4 Lze pozastavit odesílání

- Při zaplněném HW
- `netif_stop_queue`
- Při volném HW později: `netif_wake_queue`

„Odesílání“ paketů

- 1 Doplněte kód pro odesílání (`start_xmit`)
 - Vypište obsah paketu (`print_hex_dump`)
 - Uvolněte `skb` (`dev_kfree_skb`)
- 2 Vložte modul do systému
- 3 Puštěte si `arping`
- 4 Pozorujte `dmesg`

Např. z přerušování

- 1 `skb = netdev_alloc_skb(dev, len)`
 - `len` např. MTU nebo přesná velikost
- 2 Naplnění daty z HW do `skb->data` o délce max. `len`
- 3 Nastavení `skb->len`
 - `skb->len ≤ len`
- 4 Nastavení protokolu
 - `skb->protocol = eth_type_trans(skb, dev)`
- 5 Předání `skb` jádru
 - `netif_rx(skb)`
 - Jádro `skb` uvolní

„Příjem“ paketů

- 1 Emulujte příjem paketů
 - V `open` nastavte periodický časovač
 - Časovač zrušte ve `stop`
 - V časovači volejte `netif_rx` s paketem z `pb173/12/`
 - Inkrementujte o 1 v paketu 24. znak
- 2 Vložte modul do systému
- 3 Spusťte `tcpdump -ni` na `eth` rozhraní