

PB173 – Ovladače jádra – Linux

XII.

Jiří Slabý

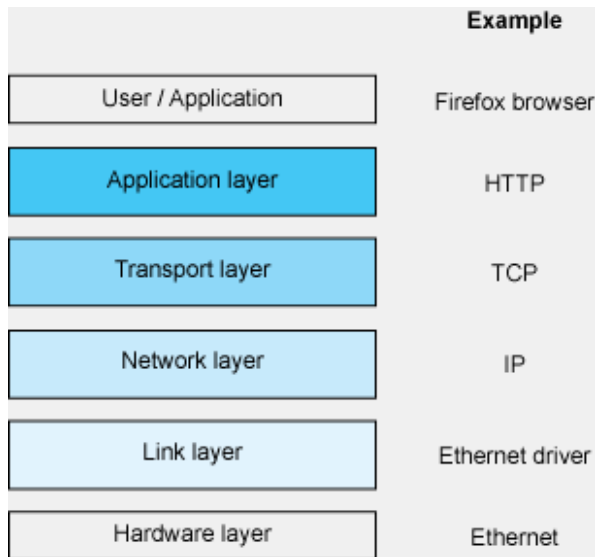
ITI, Fakulta Informatiky

14. 12. 2010

LDD3 kap. 17

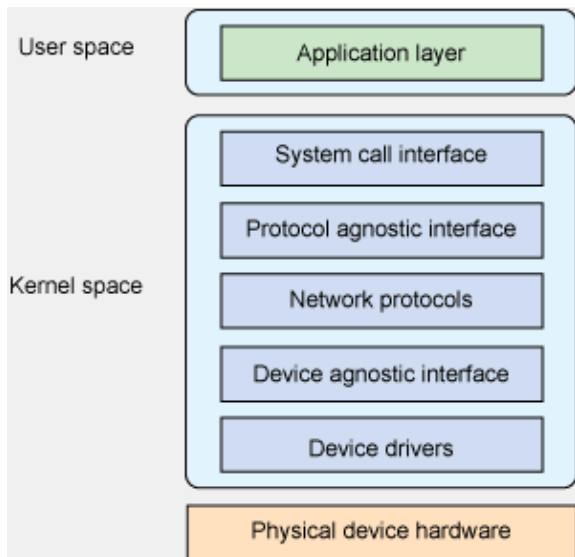
- Ovladač pro ethX
- Poslední cvičení

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*` apod. 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
 - Staré jádro: v `netdevice` přímo `open`, `stop`, ...
 - Nové jádro: `net_device_ops` (podobně jako znaková zařízení)

```
struct net_device_ops {
    int (*ndo_open)(struct net_device *dev); /* ifup */
    int (*ndo_stop)(struct net_device *dev); /* ifdown */
    netdev_tx_t (*ndo_start_xmit)(struct sk_buff *skb,
        struct net_device *dev); /* send packet */
    int (*ndo_change_mtu)(struct net_device *dev,
        int new_mtu); /* change MTU */
    ... /* no receive packet */
};
```

- Místo `alloc_netdev` se použije konkrétnější
- Nastaví vnitřní položky `netdevice`
 - *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)`)
- 2 Vyplnění informací
 - Háčky (`net_device_ops`)
 - MAC adresa (`dev->dev_addr`)
 - Získání privátního ukazatele o velikosti `priv_size` (`netdev_priv`)
- 3 Registrace (`register_netdev`)
- 4 ...
- 5 Deregistrace (`unregister_netdev`)
- 6 Uvolnění (`free_netdev`)

Vytvořit eth*

- 1 Dle postupu z předchozího slidu
 - Registrace v `module_init`
 - Deregistrace v `module_exit`
 - MAC: `random_ether_addr`
 - `priv_size: sizeof(struct timer_list) - setup_timer`
- 2 Vytvořit `open`, `stop`, `start_xmit`
 - Těla vypíšu jen název funkce
 - `start_xmit` vrátí `NETDEV_TX_OK`
- 3 Vložit do systému, udělat `ifup`, `ifdown`, `ping`

Tzv. socket buffery

- **linux/skbuff.h**, struct sk_buff
- dev_alloc_skb_D (obecně), netdev_alloc_skb_D (pro netdevice), dev_kfree_skb_D
- Obsah: skb->data
- Délka obsahu: skb->len
- Příjem z HW (např. z přerušení)
 - skb = netdev_alloc_skb(dev, len)
 - Naplnění daty z HW
 - Nastavení protokolu (skb->protocol = eth_type_trans(skb, dev))
 - netif_rx(skb)
- Posílání do HW (parametr ve start_xmit)
 - Poslat paket
 - dev_kfree_skb(skb)

Posílání/příjem paketů

- 1 Doplnit kód TX
 - Vypsát obsah paketu (`print_hex_dump_bytes`)
 - Uvolnit `skb`
- 2 Nastavit RX
 - V `open` nastavit periodický časovač, smazat v `stop`
 - V časovači volat `netif_rx` s paketem z `pb173/12`
 - Inkrementovat v paketu 24. char
- 3 Vložit module do systému
- 4 Spustit `tcpdump -ni` na `eth` rozhraní