# PB173 – Binární programování Linux

## XI. Komunikace s HW

Jiri Slaby

Fakulta informatiky
Masarykova univerzita

29. 11. 2016

# Komunikace s HW

- Obvykle práce ovladačů v jádře
- Ale často není jádro třeba
  - Zařízení na sériovém portu
  - Skenery, tiskárny a podobná USB zařízení
  - I jednoduchá PCI zařízení

# Různé přístupy ke komunikaci s HW

1. Jádro HW ovládá a uživatelovi nevystavuje
   - V Linuxu téměř nic
2. Jádro zaštiťuje nízkoúrovňovou komunikaci
   - Zařízení v `/dev/` (`ttyS*`, `ttyUSB*`, `lp*`)
   - Soubory v `/sys/` (gadgety)
3. Jádro obsluhuje jen přerušení
   - Vrstva UIO
   - Dokumentace: uio-howto
4. Jádro nedělá s HW nic
   - Uživatel používá instrukce/volání
     - Získání povolení: `ioperm` nebo `iopl`
     - Přímý přístup: `out+in`
   - Soubory v `/sys/` (PCI)
     - `mmap`
   - Knihovny na přímou komunikaci s HW
     - `libusb`, `libpciaccess`

# Přístup instrukcemi/voláními

## I/O porty

- Porty jsou specifikum některých CPU (včetně `x86`)
  - Samostatná (malá) sběrnice
  - Speciální instrukce (`in`, `out` na `x86`)
  - Je zde řadič klávesnice, PC spkr, časovače a ovladače přerušení, ladicí port (`0x80` $\Rightarrow$ segmentový displej na desce), ...
- Žádost o přístup: `ioperm`, `iopl`
- Čtení: `inX`
- Zápis: `outX`, kde X $\in \{$`b`, `w`, `l`$\}$
- Dokumentace: `man ioperm` a `man inb`

# Úkol

**Zahrajte melodii na PC speaker** (doplňte `pb173-bin/11/speaker.c`)

1. Nastartujte si virtuální stroj (`pb173-bin/qemu-start`)
2. Dopište tělo funkce `beep`
   - Zapište `0xb6` na port `0x43`
   - Zapište `div` na port `0x42`
   - Zapište `(div >> 8)` na port `0x42`
   - Přečtěte port `0x61` (obsah spodních 2 bitů zahoďte)
   - Zapište přečtenou hodnotu `|` `0x03` zpět na port `0x61`
3. Dopište tělo funkce `stop_beep`
   - Přečtěte port `0x61`
   - Zapište `0` na spodní 2 bity portu `0x61`
4. Po všech zápisech a čteních čekejte alespoň 10 $\mu$s
   - Máte k dispozici `usleep`
5. Přidejte do `main` před `play` volání `ioperm`
6. Spusťte

# PCI zařízení

- PCI, PCI-X, PCIe
- Hierarchická sběrnice
  - Identifikace doména:bus:slot.funkce
  - Bridge (=routery)
- Konfigurační prostor
  - Automatická konfigurace
  - ID zařízení (vendor, device), I/O prostory, IRQ
    - *Obsah* I/O prostorů – specifikace zařízení (výrobce)
  - `lspci`
- Podrobnosti v PCI specifikaci

# `mmap` pro přímou komunikaci s PCI

- Každé PCI zařízení může mít:
  - I/O porty (komunikace viz dříve)
  - *Paměť ve fyzickém prostoru*
  - Přerušení, DMA, atd. (UIO)
- Soubory `/sys/bus/pci/devices/*/resource*`
- Mapují se pomocí `mmap`
- Obsah: ve specifikaci konkrétního zařízení

# Úkol

## Komunikace s EDU zařízením

1. Zjistěte číslo EDU karty z výstupu `lspci`
   - Má ID `0x11e8` v `qemu`
   - Hledáte něco jako `01:01.0`
2. Mapujte `resource0` EDU zařízení
   - V `/sys/bus/pci/devices/...`
   - Jen v qemu s `-device edu`
   - Mapujte jako `volatile uint32_t *`
3. Vypište, co je na první a druhé pozici
4. Zapište nějaké číslo na druhou pozici
5. Vypište, co je na druhé pozici
6. Spusťte

# Knihovny pro přímou komunikaci s HW

- Kromě přímého `mmap` přístupu existují i knihovny
- `libusb`, `libpciaccess`
  - Jsou přenositelné
  - Poskytují lepší rozhraní
  - Pracují se soubory v `/proc` a `/sys`

# libpciaccess

- `gcc ... -lpciaccess` (`pciaccess.h`)
- Inicializace: `pci_system_init`
- Deinicializace: `pci_system_cleanup`
- Vytvoření iterátoru (`struct pci_device_iterator *`)
  - Vím ID: `pci_id_match_iterator_create` a `struct pci_id_match`
  - Vím slot: `pci_slot_match_iterator_create` a `struct pci_slot_match`
- Procházení iterátoru: `pci_device_next`
  - Vrací `struct pci_device *` (a `NULL` nakonec)
- Zničení iterátoru: `pci_iterator_destroy`

## Struktury `pci_id_match` a `pci_slot_match`

```
uint32_t vendor_id;
uint32_t device_id;
uint32_t subvendor_id;
uint32_t subdevice_id;
```

```
uint32_t domain;
uint32_t bus;
uint32_t dev;
uint32_t func;
```

PCI_MATCH_ANY

# Úkol

## Výpis PCI zařízení pomocí `libpciaccess`

1. Volejte `pci_system_init`
2. Vytvořte iterátor
   - S parametrem `NULL`, tj. projdi všechny
3. Projděte zařízení
4. Pro každé vypište číslo
   - Domény
   - Sběrnice
   - Zařízení
   - Funkce
5. Spusťte a zkontrolujte s `lspci -nn`

# libpciaccess a zařízení

- Před další prací je nutné inicializovat `struct pci_device`
  - Pomocí `pci_device_probe`
  - Potom lze pracovat s dalšími prvky `struct pci_device`
    - `vendor_id`, `device_id`, `irq`, ...
- Mapování regionu: `pci_device_map_range`
  - Mapuje region do paměti
  - Podobné `mmap` regionu z dnešního cvičení
- Konfigurační prostor
  - Zapisuje BIOS a OS
  - Informace o zařízení
  - Čtení: `pci_device_cfg_read*`
  - Zápis: `pci_device_cfg_write*`

# Úkol

**Práce s EDU pomocí `libpciaccess`**

1. Mapujte si 0. region
   - `pci_device_map_range`
   - Mapujte jako `volatile uint32_t *`
   - Podobně jako `mmap` předtím
2. Vypište, co je na první a *třetí* pozici
3. Zapište nějaké malé číslo na třetí pozici
4. Počkejte, než bude osmá pozice `0`
5. Vypište, co je na třetí pozici
6. Spusťte