PB173 – Binární programování Linux

X. Komunikace s HW

Jiri Slaby

ITI, Fakulta informatiky

26. 11. 2013

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

Principy komunikace s HW

1. Jádro zaštiťuje nízkoúrovňovou komunikaci
   - Zařízení v `/dev/` (`ttyS*`, `ttyUSB*`, `lp*`)
   - Soubory v `/sys/` (gadgety)
2. Jádro ovládá jen přerušení
   - Vrstva UIO
   - Dokumentace: uio-howto
3. Jádro nedělá s HW nic
   - Systémová volání
     - `ioperm`, `out+in`
   - Soubory v `/sys/` (PCI)
     - `mmap`
   - Knihovny pro přímou komunikaci s HW
     - `libusb`, `libpciaccess`

Systémová volání – 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

1. Implementujte si funkci `beep(unsigned int hz)`
   - Zapište `0xb6` na port `0x43`
   - Zapište `0x1234dd/hz` na port `0x42`
   - Zapište `(0x1234dd/hz)>> 8` na port `0x42`
   - Přečtěte port `0x61` (obsah spodních 2 bitů zahoďte)
   - Zapište `0x03` na port `0x61`
2. Implementujte si funkci `stop_beep(void)`
   - Přečtěte port `0x61` (obsah zahoďte)
   - Zapište `0x00` na port `0x61`
3. Po všech zápisech a čteních čekejte alespoň 10 μs
4. Zavolejte `ioperm`
5. Zavolejte `beep` s různými frekvencemi
   - Přidejte čekání za každý beep, aby tón chvíli zněl
6. Zavolejte `stop_beep`
7. Spusťte

PCI zařízení

- PCI, PCI-X, PCIe
- Hierarchická sběrnice
  - Identifikace doména:bus:slot:funkce
  - Bridge (=routery)
- Konfigurační prostor (ROM)
  - Automatická konfigurace
  - ID zařízení (vendor, device), I/O prostory, IRQ
    - Obsah I/O – 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. Mapujte `resource0` PCI zařízení s ID `0x11e8` v `qemu`
   - Jen v laboratoři na virtuálních strojích
   - Mapujte jako `volatile unsigned int *`
2. Vypište, co je na první a druhé pozici
3. Zapište nějaké číslo na druhou pozici
4. Vypište, co je na druhé pozici
5. Spusťte

Knihovny pro přímou komunikaci s HW

- Kromě `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`
  - Vím slot: `pci_slot_match_iterator_create`
- Procházení iterátoru: `pci_device_next`
  - Vrací `struct pci_device *` (a `NULL` nakonec)
- Zničení iterátoru: `pci_iterator_destroy`

Úkol

Výpis zařízení pomocí `libpciaccess`

1. Volejte `pci_system_init`
2. Vytvořte iterátor (parametr `NULL`)
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/je v ROM
  - 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 unsigned int *`
   - 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