

PB173 – Binární programování Linux

XI. Komunikace s HW

Jiri Slaby

Fakulta informatiky
Masarykova univerzita

10. 12. 2015

- 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 Jádru HW ovládá a uživateli nevystavuje
 - V Linuxu téměř nic
- 2 Jádru zaštiťuje nízkourovňovou komunikaci
 - Zařízení v `/dev/` (`ttyS*`, `ttyUSB*`, `lp*`)
 - Soubory v `/sys/` (`gadgety`)
- 3 Jádru obsluhuje jen přerušení
 - Vrstva UIO
 - Dokumentace: `uio-howto`
- 4 Jádru 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`

- 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 přerušování, 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`

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 0x1234dd/hz na port 0x42
 - Zapište (0x1234dd/hz)>> 8 na port 0x42
 - Přečtěte port 0x61 (obsah spodních 2 bitů zahod'te)
 - 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 μ s
 - Máte k dispozici usleep
- 5 Přidejte do main volání ioperm
- 6 Spus'te

- 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 – specifikace zařízení (výrobce)
 - lspci
- Podrobnosti v PCI specifikaci

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

Komunikace s EDU zařízením

- 1 Mapujte `resource0` PCI zařízení s ID `0x11e8` v qemu
 - Jen v qemu s `-device edu`
 - Mapujte jako `volatile uint32_t *`
- 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

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

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

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`

- 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*`

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