

PB173 – Ovladače jádra – Linux

VII.

Jiří Slabý

ITI, Fakulta Informatiky

9. 11. 2010

LDD3 kap. 9 a 12

- I/O porty a paměť
 - Tj. samotná komunikace
- Práce s PCI zařízeními
- Zobecnění na jiné sběrnice

Memory Mapped I/O

- Součástí fyzického adresového prostoru
- Přístup standardním čtením/zápisem
 - Nutnost přemapovat na virtuální adresy

API

- `linux/io.h`, `linux/ioport.h`
- Vytvoření: `request_mem_region`, `release_mem_region`
 - `/proc/iomem`
- Mapování: `virt=ioremap(phys)`, `iounmap(virt)`
 - `/proc/vmallocinfo` (novější jádra)
- R/W: `readX(odkud)`, `writeX(co, kam)`, kde $X \in \{b, w, l, q\}$
- Vícenásobné R/W: `memcpy_fromio`, `memcpy_toio`

Úkol: Přečíst ACPI identifikaci jako minule, ale s použitím těchto funkcí a kontrolou `/proc` (př.: `drivers/misc/arm-charlcd.c`)

Porty

- Speciální adresový prostor
 - Závislé na architektuře
 - Speciální instrukce (`in`, `out` na x86)
- Samostatná (malá) sběrnice
 - Na x86: řadič klávesnice, PC spkr, staré časovače a ovladače přerušení, ladicí port (0x80 \Rightarrow segmentový displej na desce), ...

API

- `linux/io.h`, `linux/ioport.h`
- Vytvoření: `request_region`, `release_region`
- R/W: `inX(port)`, `outX(co, port)`, kde $X \in \{b, w, l\}$
- Vícenásobné R/W: `insX`, `outsX`

Demo: pb173/07

Úkol: Přečíst port 0x80, zapsat do něj 1B a znovu přečíst

Respektujte soukromí

pb173/marktwain

Pro porty a MMIO

- Pomalejší (jeden `if` při každém přístupu)
- Mapování: `*_iomap *_iounmap (pci_iomap)`
- R/W: `ioreadXX, iowriteXX`, kde $X \in \{8, 16, 32, 64\}$

**Většina rozhraní implementuje MMIO,
porty spíše vyjímečně**

- 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

PCI zařízení v jádře I.

- `linux/pci.h`, `Documentation/pci/*`
- `struct pci_dev`, `struct pci_bus`
- `pci_{set, get}_drvdata` – uloží/načte programátorova data
- `PCI_ANY_ID` značí jakékoliv ID

Hledání zařízení

- 1 Iterátory (starší)
 - Vícenásobný přístup k zařízení
 - Nepodporuje hotplug
 - `pci_get_device (vendor, device)`
 - Reference: `pci_dev_get`, `pci_dev_put`

```
struct pci_dev *pdev = NULL;
while ((pdev = pci_get_device(VENDOR, DEVICE, pdev))) {
    printk ("%2.x:%.2x:%.2x\n", pdev->bus->number, PCI_SLOT(pdev->devfn),
            PCI_FUNC(pdev->devfn));
}
```

Úkol: vypsát všechna zařízení v systému (jejich ID)

PCI zařízení v jádře II.

2 Událostmi

- Registrace seznamu chtěných zařízení a háčeků
- Seznam: `struct pci_device_id (vendor, device, atd.)`
- Háčky: `struct pci_driver (probe, remove, suspend, atd.)`
- `pci_register_driver, pci_unregister_driver`

```
struct pci_device_id my_table[] = {
    { PCI_DEVICE(VENDOR1, DEVICE1) }, { PCI_DEVICE(VENDOR1, DEVICE2) },
    { PCI_DEVICE(0x8086, PCI_ANY_ID), .driver_data = 1 },
    { 0, }
};
MODULE_DEVICE_TABLE(pci, my_table);
```

```
struct pci_driver my_pci_driver = {
    .name = "my_driver",      .id_table = my_table,
    .probe = my_probe,      .remove = my_remove,
};
```

```
int my_probe(struct pci_dev *pdev, const struct pci_device_id *id)
{
    printk("%2.x %lu\n", pdev->bus->number, id->driver_data);
    return 0;
}
```

Navázání PCI zařízení

- 1 **linux/pci.h**
- 2 Definice struct `pci_device_id`
 - `lspci -nn` a najít Combo a jeho vendor+device ID
- 3 Definice struct `pci_driver`
 - `probe`, `remove`, `name`, `id_table`
- 4 Definice háčeků (viz definici struct `pci_driver`)
 - `int (*probe)(struct pci_dev *, const struct pci_device_id *)`
 - `void (*remove)(struct pci_dev *)`
- 5 V `probe` a `remove` vypsát
 - `pdev->bus->number`
 - `PCI_SLOT(pdev->devfn)`
 - `PCI_FUNC(pdev->devfn)`

Probe

- Inicializace PCI zařízení
 - `pci_enable_device`
 - Do té doby nelze některé vlastnosti pdev používat (`irq`)
- Rezervace a mapování I/O (první část cvičení)
 - Vytvoření: `pci_request_region`, `pci_request_regions`
 - Mapování: `pci_ioremap_bar` – alias pro
`ioremap(pci_resource_start(), pci_resource_len())`
- Nastavení/detekce zařízení

Remove

- Opak Probe
- `iounmap`, `pci_release_region(s)`, `pci_disable_device`

Většina ostatních sběrnic funguje stejně

- USB, I²C, HID, IEEE1394, ACPI, INPUT, EISA, . . .
- Nějaké probe/remove
- Seznam ID
- Registrace „ovladače”

Vyčtení identifikace

- 1 Povolení zařízení (`pci_enable_device`)
- 2 Rezervace baru 0 (`pci_request_region`)
- 3 Vypsát fyzickou adresu baru 0 a porovnat s `lspci`
- 4 Přemapovat bar 0 (`ioremap`)
- 5 Přečíst a rozkódovat identifikaci dole (`readX`)
- 6 Uklidit v remove (`unmap`, `release`, `disable`)

Offset	Len	R/W	Contents	Meaning
0x0000	4B	R	0xC610RRrr	Design ID & Revision
0x0004	4B	R	0xYMDDhhmm	Design build time

Tabulka: Specifikace baru 0 karet COMBO

Pozn.: součást domácího