

PB173 – Ovladače jádra – Linux X.

Jiří Slabý

ITI, Fakulta Informatiky

29. 11. 2011

LDD3 kap. 15 (zastaralá)

- I. (minule): Mapování paměti jádra (mmap)
- II. (dnes): Přímý přístup do paměti (DMA)

Přímý přístup do paměti

- Prozatím jsme ze/do zařízení četli/zapisovali přes CPU
 - Tj. standardními operacemi s (přemapovanou) pamětí
- Velké přenosy = velká zátěž CPU
 - Cykly, čekání na pomalou sběrnici atd.
- Místo toho naprogramujeme HW, aby přenášel data sám
 - Nutná podpora HW (sběrnice – arbitrace, zařízení – přenosy)

Princip

- 1 Alokace (speciální) paměti
- 2 Pro TX: vyplnění daty (paket, zvuk, data na disk, ...)
- 3 Předání ukazatele do zařízení
- 4 Odstartování přenosu v zařízení
- 5 Přerušování či jiná signalizace konce přenosu od zařízení
- 6 Pro RX: práce s příchozími daty (paket, data z disku, ...)

API alokace

- `linux/dma-mapping.h`, `Documentation/DMA-*`
- `dma_alloc_coherent`, `dma_free_coherent`

```
void *dma_alloc_coherent(struct device *dev, size_t size,  
dma_addr_t *dma_handle, gfp_t gfp)
```

- `dev` – zařízení, které bude k paměti přistupovat (NULL == neznáme)
- `size` – velikost, jakou požadujeme
- `dma_handle` – fyzická adresa (návratová hodnota) – pro zařízení
- návratová hodnota – virtuální adresa – pro nás

DMA v Linuxu – příklad

```
int my_do_DMA(struct device *dev)
{
    dma_addr_t phys;
    void *virt;

    virt = dma_alloc_coherent(dev, 100, &phys, GFP_KERNEL);
    if (!virt)
        return -ENOMEM;
    memset(virt, 0, 100);
    my_HW_set_addr(dev, phys);
    my_HW_start_transfer(dev);
    while (my_HW_working(dev)) /* polling */
        msleep(100);
    printk(KERN_INFO "%s", virt);
    return 0;
}
```

Úkol: alokujte pomocí `dma_*` 1 stránku paměti (pro `dev = NULL`)

Navíc

- Nastavení masky adres
 - Ne všechna zařízení zvládnou adresovat celý fyzický prostor
 - `pci_set_dma_mask(pdev, DMA_BIT_MASK(27))`
- Zapnutí „spravování sběrnice“
 - Tj. zařízení umí samo iniciovat přenosy apod.
 - `pci_set_master(pdev)`

```
int my_probe(struct pci_dev *pdev, ...)  
{  
    dma_addr_t phys;  
    void *virt;  
    ...  
    ret = pci_set_dma_mask(pdev, DMA_BIT_MASK(27));  
  
    pci_set_master(pdev);  
  
    virt = dma_alloc_coherent(&pdev->dev, 100, &phys, GFP_KERNEL);  
    ...  
}
```

- Zvládá 32-bitové adresy
- DMA řadič napojený na LocalBus (1), PCI (2) a PowerPC (4)
- Generuje přerušení 8 na kartě

Úkol: upravte předešlou alokaci (`set_dma_mask`, `set_master`, `dev`)

Offset	Len	R/W	Contents	Meaning
0x0080	4B	R/W	src	Source address
0x0084	4B	R/W	dst	Destination address
0x0088	4B	R/W	count	Transfer count
0x008c	4B	R/W	cmd	Command register

Tabulka: Specifikace baru 0 (pokračování z minula)

- cmd registr
 - Bit 0: RUN (transfer now)
 - Bits 1-3: SOURCE BUS
 - Bits 4-6: DESTINATION BUS
 - Bit 7: NOINT (do not generate interrupt after transaction)
 - Bit 31: ACKINT (acknowledge interrupt)

Práce s DMA

- 1 Naalokovat stránku DMA prostoru
- 2 10B inicializovat textovým řetězcem
- 3 Přenést 10B na PowerPC sběrnici na adresu 0x40000
 - Nastavit všechny 4 registry
 - Bez přerušení (nastavit `NOINT`)
 - Počkat na nulovou hodnotu bitu `RUN` (na dokončení přenosu)
- 4 Přenést 10B z PowerPC na DMA stránku + 10
 - Vznikne za sebou 2× stejný řetězec

DMA s přerušením (domácí)

- 1 Rozšířit předchozí o přerušení
 - Přidat ještě jeden přenos z PowerPC na DMA + 20
 - Přenos bude bez `NOINT`
 - Je třeba povolit 8. (0x100) přerušení na kartě
- 2 Obsloužit přerušení
 - ACK přerušení a DMA
 - Iniciovat tasklet
- 3 V taskletu vypsát obsah DMA paměti + 20
- 4 DMA stránku vystavit přes mmap

Pozn.: pokud se zařízení zasekne (např. špatně nastaveným DMA), je nutné stroj vypnout a zapnout, reboot nepomůže