Operačné systémy / Vstupno-výstupný systém

Dušan Bernát
bernat@fmph.uniba.sk

Prehľad

- Typy vstupno/výstupnej komunikácie,
- Prístup procesu k periférnym zariadeniam,
- Špeciálne súbory zariadení,
- Znakové zariadenia,
- Blokové zariadenia,
- GPIO.

Typy riadenia V/V komunikácie

Periférne zariadenia

- Zjednodušene: všetko okrem CPU a hlavnej pamäte.
  - Terminál, disky, sieťové rozhrania, modemy, tlačiarne, zvukové prevodníky, kamery, GPS, senzory, spínače, ...
  - Informácie o perifériách (Linux): **`lshw`**, **`lspci`**, **`lsusb`**, **`lsblk`**, **`dmidecode`**
- Operačný systém komunikuje s periférnymi zariadeniami.
  - Posiela požiadavky, reaguje na ich podnety, obsluhuje prerušenia, ošetruje chyby.
  - Časť ktorá to zabezpečuje sa nazýva ovládač zariadenia. Je hardvérovo špecifický.
- Poskytuje procesom rozhranie pre prístup k perifériám.
  - Toto rozhranie by malo byť v čo najväčšej miere pre všetky zariadenia rovnaké → abstrakcia.

Prístupy k riadeniu V/V komunikácie

- Programové V/V operácie.
  - Pošle požiadavku a potom (aktívne) čaká na odpoveď, alebo zmenu stavu.
  - Procesor je využívaný počas celej komunikácie so zariadením (polling / busy waiting).
- V/V operácie riadené prerušením.
  - Procesor pošle požiadavku a môže robiť niečo iné. Proces môže byť uspaný.
  - Periféria vygeneruje prerušenie keď sú dáta dostupné, alebo zmení stav.
  - Po prerušení môže OS znovu naplánovať proces komunikujúci so zariadením. Vyhne sa tak obsadzujúcemu čakaniu.
- V/V s využitím DMA (*Direct Memory Access*).
  - Keď má zariadenie pripravené dáta, presunie ich priamo do pamäte. Zbernicu riadi radič DMA. Potom vygeneruje prerušenie.
  - Procesor je teda potrebný až keď sú dáta už prenesené v pamäti.

Adresovanie periférií

- V/V (I/O) porty – signály pre komunikáciu s perifériami.
  - Obvody periférnych zariadení majú väčšinou vyhradenú riadiacu časť (*controller*).
    - Obsahuje riadiace registre, zápis do nich môže predstavovať príkaz, alebo priamo nastaviť signály. Čítanie sprístupňuje stav.
    - Dátové registre (*buffer*) na prenos dát.
- **Pamäťovo mapované** – ovládajú sa prístupom na vyhradené adresy v pamäťovom priestore (MOV, LOAD/STORE).
- **Samostatné** – zvláštny V/V adresový priestor (0 - 65535), zvláštne inštrukcie (IN/OUT).
  - x86: in{bwl}, out{bwl}
- Informácie (Linux): `/proc/iomem`, `/proc/ioports`, `/proc/interrupts`

Privilégiá prístupu k perifériám

- Kto môže vykonať V/V inštrukcie?
  - Jadro (ring0) - áno
  - Proces (ring3) - ?
- Procesor pri vykonávaní V/V inštrukcií kontroluje nielen CPL (*ring*), ale aj IOPL - *Input/Output Privilege Level* (2 bity).
- Kto môže zmeniť IOPL? Priamo len jadro.
- Proces môže požiadať jadro o nastavenie IOPL bitov. Systémové volania:
  - `int iopl(int level);`
  - `int ioperm(unsigned long from, unsigned long num, int turn_on);`
  - Môže ich vykonať len používateľ *root* (ten to môže delegovať aj na iných).
- Knižničné funkcie (makrá):
  - `inb()`, `outb()`, ...

Procesy a periférie

- Priamy prístup procesov k hardvéru je možný (ale väčšinou nie je žiaduci).
  - Aplikácie pracujúce so senzormi a pod. (vnorené systémy).
  - Jednoúčelové, nevyžadujú (zložitý) vývoj ovládača.
- Ak má proces prístup ku všetkým V/V zariadeniam, môže systém poškodiť (radiče prerušení, diskov, ...).
  - Umožnilo by to napríklad obchádzať riadenie prístupu OS k prostriedkom.
- Proces nemôže obsluhovať prerušenia.
- Operačný systém môže niektoré úlohy prenechať procesom (jadro zostáva kompaktné, mikrokernel).
  - Ovládače zariadení v používateľskom režime. Ak ovládač “spadne”, zvyšok systému to neovplyvní.

Rozhranie zariadení

Zariadenia ako súbory

- Unix – všetko je súbor. S perifériami môžu procesy komunikovať pomocou “súborových” operácií.
- Súbory typu ’c’ a ’b’ predstavujú znakové a blokové zariadenia. Sú spravidla umiestnené v adresári `/dev`.
- Nie sú spojené s diskom, nemajú dátové bloky.
- Sú spojené s ovládačom zariadenia v jadre, ktorý je určený tzv. hlavným číslom zariadenia (*major number*).
- Viaceré zariadenia rovnakého typu, obsluhované rovnakým ovládačom, sú rozlíšené tzv. vedľajším číslom (*minor number*).
  - Pri vykonávaní operácie ho ovládač použije na identifikovanie konkrétneho zariadenia.
- Napríklad:
  - Diskové oddiely jedného disku majú rovnaké hlavné číslo a rôzne vedľajšie čísla.

Príklad: Špeciálne súbory zariadení


```
$ls -l /dev
crw-r-----. 1 root kmem   1,  1   Feb 24 10:14 mem
crw-rw-rw-. 1 root root   1,  3   Feb 24 10:14 null
crw-r-----. 1 root kmem   1,  4   Feb 24 10:14 port
crw-rw-rw-. 1 root root   1,  8   Feb 24 10:14 random
brw-rw----. 1 root disk   8,  0   Feb 24 10:14 sda
brw-rw----. 1 root disk   8,  1   Feb 24 10:14 sda1
brw-rw----. 1 root disk   8,  2   Feb 24 10:14 sda2
brw-rw----. 1 root root   8, 16   Feb 24 10:14 sdb
brw-rw----. 1 qemu qemu   8, 17   Mar 16 10:09 sdb1
crw-rw-rw-. 1 root tty    5,  0   Mar  9 11:49 tty
crw--w----. 1 root tty    4,  0   Mar  9 11:49 tty0
crw--w----. 1 root tty    4,  1   Mar  9 11:49 tty1
crw-rw-rw-. 1 root root   1,  5   Feb 24 10:14 zero
```

Príklad: Ovládače zariadení

```
$cat /proc/devices

Character devices:
  1 mem
  4 tty
  5 /dev/tty
  5 /dev/console
 14 sound
 21 sg
180 usb
188 ttyUSB

Block devices:
  7 loop
  8 sd
  9 md
 11 sr
```

Ovládač `mem`

- Poskytuje viacero funkcií, rozlišujú sa vedľajším číslom.
- `1 mem` – obraz fyzickej pamäte.
- `2 kmem` – obraz virtuálnej pamäte jadra.
- `4 port` – prístup k I/O portom.
- `3 null` – zápis vždy uspeje, dáta sa zahodia, čítanie vráti EOF.
- `5 zero` – zápis vždy uspeje, dáta sa zahodia, čítanie vráti nuly.
- `7 full` – zápis vždy zlyhá (ENOSPC), čítanie vráti nuly.
- `8 random`, `9 urandom` – čítanie vráti náhodné hodnoty, zápis aktualizuje entrópiu. Pokiaľ entrópie nie je dostatok, `urandom` použije pseudonáhodný generátor, `random` čítanie zablokuje.

Registrácia ovládača

- Ovládač môže byť pevnou súčasťou jadra, alebo modul.
- Pri inicializácii zaregistruje svoje číslo (*major number*) a operácie pre špeciálny súbor.

```
struct file_operations new_file_ops;
register_chrdev(NEW_MAJOR, "new", &new_file_ops);
```

- Jednotlivé špeciálne súbory zariadení (s rôznymi *minor number*) môžu byť vytvorené:
  - Napevno, volaním **`mknod()`**, **`mknod /dev/new c 200 0`**
  - alebo ovládačom, keď pri inicializácii deteguje prítomnosť zariadenia, funkciou **`device_create()`**.

Typ zariadenia

- Všetky periférie sú rozdelené do dvoch skupín.
- **Znakové zariadenia** spravidla komunikujú postupnosťou bajtov (stále nové dáta, teda žiadna *cache*).
- **Blokové zariadenia** musia poskytnúť požadovaný blok dát (*random access*). Využívajú tzv. *buffer cache*.
  - Ovládač neimplementuje operácie `read()` a `write()`, namiesto toho poskytuje tzv. funkciu “stratégie”.
  - Funkcia stratégie spracováva zoznam požiadaviek na čítanie a zápis blokov, robí všetky potrebné V/V operácie.

Blokové zariadenia

Funkcia stratégie

- Jadro používa funkcie **`bread()`** a **`bwrite()`** na operácie s blokmi cez *buffer cache*.
- Ak blok nie je k dospozícii, zavolá funkciu stratégie.
- Funkcia stratégie spracováva zoznam požiadaviek na čítanie a zápis blokov, robí všetky potrebné V/V operácie (*polling*, *interrupt driven*).
  - Funkcia nemá žiadny vstup, ani výstup. Vie kde je zoznam požiadaviek.
  - Volá sa so zakázanými prerušeniami, pred koncom musí urobiť **`sti()`**.
  - Môže využívať *polling* (čaká na dokončenie požiadavky zariadením), alebo prerušenia.
  - Pri využití prerušení len pošle požiadavku radiču. Pri obsluhe prerušenia sa požiadavka odoberie zo zoznamu a funkcia stratégie sa zavolá znova.
  - Požiadavka sa odoberie zo zoznamu aj v prípade (opakovaného) zlyhania.

Organizácia dát na disku

- Celková doba prístupu k diskovým blokom môže byť závislá od poradia prístupov.
  - Diskový priestor je organizovaný v sektoroch ktoré sú umiestnené na stopách (sútredné kružnice). Všetky stopy umiestnené nad sebou (s rovnakým polomerom) tvoria tzv. cylinder.
  - Hlava sa nastavuje na stopu, potom môže vyhľadať a čítať sektory.
  - Cylindre s nižšími číslami (menším polomerom) majú väčšie rýchlosti prenosu.
  - Je potrebné minimalizovať pohyby mechanických častí (čítacích hláv).
- Funkcia stratégie pozná všetky aktuálne požiadavky a môže ich zoradiť podľa vlastností konkrétneho disku.

Optimalizácia prístupu k disku

- *First-Come, First-Serve* (FCFS) – Obsluhujú sa v poradí v akom prišli.
- *Elevator algorithm* (SCAN) – Hlava sa hýbe od jedného konca disku k druhému (ako výťah, hore-dolu). Požiadavky sa vybavujú v poradí čísiel cylindrov na ktorých sa bloky nachádzajú. Na konci sa otočí. Algoritmus musí poznať aktuálnu pozíciu a smer pohybu hlavy.
- *Circular SCAN* (C-SCAN) – Požiadavky sa obsluhujú len pri pohybe jedným smerom. Keď príde na koniec, prejde na začiatok bez čítania. Priemerná doba čakania je rovnomernejšia. Predchádzajúce prístupy uprednostňovali cylindre umiestnené v strede.
- Oba algoritmy je možné modifikovať tak, že hlava nepôjde vždy na koniec, ale len k poslednej aktuálnej požiadavke v danom smere.
- *Shortest Seek first* (SSF) – Obslúži sa vždy požiadavka najbližšie k pozícii hlavy. Minimalizuje priemernú dobu prístupu. Požiadavky na krajné (vzdialené) cylindre môžu byť predbiehané (starvácia).
- Požiadavky na minimálnu dobu prístupu a spravodlivosť obsluhy sú v protiklade.
- Niektoré dnešné disky preusporiadavajú požiadavky samy, OS nemusí.

Princípy plánovania diskových operácií (Linux)

- Jadro udržuje rad požiadaviek pre každý proces a každý rad má pridelený čas na prístup.
  - Zaručuje to spravodlivosť.
- *Deadline* – Každá požiadavka má priradený časový limit (*deadline*). Algoritmus pracuje ako C-SCAN. Ak sa blíži *deadline* nejakej požiadavky, obslúži sa hneď.
- Algoritmus môže aj počkať na príchod ďalších požiadaviek a zahrnúť ich do optimalizácie. Ak sa počas čakania vyskytne bližšia požiadavka, obslúži sa. Inak pracuje ako C-SCAN.
  - Čakanie predlžuje dobu prístupu niektorých požiadaviek. Súhrnný čas môže byť nižší.

Operácie ovládačov zariadení

Operácie zariadenia

- Štruktúra **`file_operations`** obsahuje pointre na funkcie ovládača.
- Znakové zariadenie: lseek(), read(), write(), readdir(), select(), ioctl(), mmap(), open(), close(), release().
- Blokové zariadenie: open(), close(), psize(), strategy().
- Ak proces vykoná operáciu nad špeciálnym súborom zariadenia, OS zavolá príslušnú operáciu ovládača (určeného jeho hlavným číslom).
- Nie všetky musia byť ovládačom implementované.

Operácia IOCTL

- IOCTL - *Input/Output Control*, slúži na ovládanie, nastavovanie alebo zisťovanie stavu zariadenia, nie na prenos dát.
- Napr. Nastaviť rýchlosť prenosu (NIC, COM, ...), zistiť stav linky, nastaviť hlasitosť, ...
- Volanie `ioctl()` má dva argumenty (tretí je deskriptor):
  - **`cmd`** – rozlišuje, akú operáciu má ovládač urobiť; v čísle IOCTL volania je zakódované či ide o čítanie alebo zápis.
  - **`arg`** – dáta (obojsmerné; nastavovaný, alebo zistený stav).

Operácia MMAP

- MMAP – *Memory Map*, umožňuje namapovať časť pamäte ovládača (teda pamäť jadra) do logického adresového priestoru procesu.
  - Nie je potrebné kopírovanie medzi priestorom jadra a procesu.
- Napr. Ak zariadenie poskytuje dáta v podobe súvislého bloku (napríklad *video frame buffer*, NIC *buffer*, ...)
  - Tieto “buffer-e” bývajú často naplňované priamo zariadením, cez DMA.
- Na druhej strane po vytvorení mapovania do používateľského priestoru nie je potrebné žiadne systémové volanie, aby proces k dátam pristupoval (pristupuje ako k pamäti).

GPIO podsystém

General Purpose Input/Output

- GPIO – *General Purpose Input/Output*
- Hardvérový podsystém procesora poskytujúci softvérovo nastaviteľné a ovládateľné signály.
- Sú dostupné najmä v procesoroch pre vnorené systémy (napr. ARM).
- Signály môžu byť nastavené ako vstupné alebo výstupné, aktívne v 0 alebo 1, môžu generovať prerušenie, ...
- Signál je identifikovaný:
  - z pohľadu HW umiestnením (banka - xx, bit - yy),
  - z pohľadu OS číslom: GPIOxx_yy, gpio = (xx-1)·32+yy

Rozhranie pre jadro (1)

- Príklad: podpora v OS Linux.
- Potrebné pre písanie ovládačov zariadení pripojených cez GPIO.

```
#include <linux/gpio.h>
```

- Alokovanie a uvolnenie signálu:

```
int gpio_request(unsigned int gpio, const char *label);
void gpio_free(unsigned int gpio);
```

- Nastavenie smeru signálu:

```
int gpio_direction_input(unsigned int gpio);
int gpio_direction_output(unsigned int gpio, int value);
```

Rozhranie pre jadro (2)

- Prečítanie a nastavenie hodnoty signálu:

```
int gpio_get_value(unsigned int gpio);
void gpio_set_value(unsigned int gpio, int value);
```

- Zapnutie generovania prerušení:

```
int gpio_to_irq(unsigned int gpio);
```

- Vytvorenie ovládacích súborov v *sysfs* (`/sys`):

```
int gpio_export(unsigned int gpio,
    bool direction_may_change);
void gpio_unexport(unsigned int gpio);
```

Rozhranie pre procesy

- Rozhranie poskytuje ovládač využívajúci (pseudo) súborový systém *sysfs*.
- Zapísaním čísla signálu do **`/sys/class/gpio/export`** sa vytvorí adresár: **`/sys/class/gpio/NN`**

```
echo NN > /sys/class/gpio/export
```

- Operáciami so súbormi v tomto adresári môže proces ovládať príslušný signál;
  - **`direction`** (*in*/*out*), **`value`** (*0*/*1*), **`edge`**, **`active_low`**
  - Deskriptor súboru **`value`** je možné použiť aj pre **`select()`**.
  - Súbor **`edge`** (*none*, *rising*, *falling*, *both*) môže nastaviť aj generovanie prerušení týmto signálom.

Príklad: Práca s GPIO

```
echo 177 > /sys/class/gpio/export

echo both > /sys/class/gpio/gpio177/edge

cat /sys/class/gpio/gpio177/value

echo "out" > /sys/class/gpio/gpio177/direction

echo 1 > /sys/class/gpio/gpio177/value

echo 177 > /sys/class/gpio/unexport
```

To je (zatiaľ) všetko