Operačné systémy

Manažment I/O

Ing. Martin Vojtko, PhD.

2024/2025

Manažment I/O

- model procesu a virtuálna pamäť sú 1. a 2. pilierom OS.
- manažment vstupu a výstupu je 3. pilierom bežného OS.
- Úlohou OS vo vzťahu k I/O je:
  - rozhranie pre konfiguráciu I/O.
  - spracovanie prerušení vyvolaných I/O.
  - spracovanie chybového výstupu.
  - zabezpečiť rozhranie medzi I/O a zbytkom výpočtového systému.
- Kód ovládajúci zariadenia je podstatnou časťou OS.

Princípy I/O

Klasifikácia I/O zariadení

Block Devices

sú zariadenia, ktoré ukladajú informácie v blokoch fixnej dĺžky. Každý blok je označený jedinečnou adresou. Veľkosť bloku je bežne v rozsahu 512B - 65kB. Akékoľvek čítanie alebo zapisovanie je realizované v jednotkách blokov. (Disk, CD, flash disk...)

Character Devices

sú zariadenia, ktoré akceptujú tok znakov. Jednotlivé znaky nemajú žiadnu pridelenú adresu a nie je možné vykonávať žiadne vyhľadávanie. (sieťová karta, sériová linka, myš, klávesnica)

Event Devices

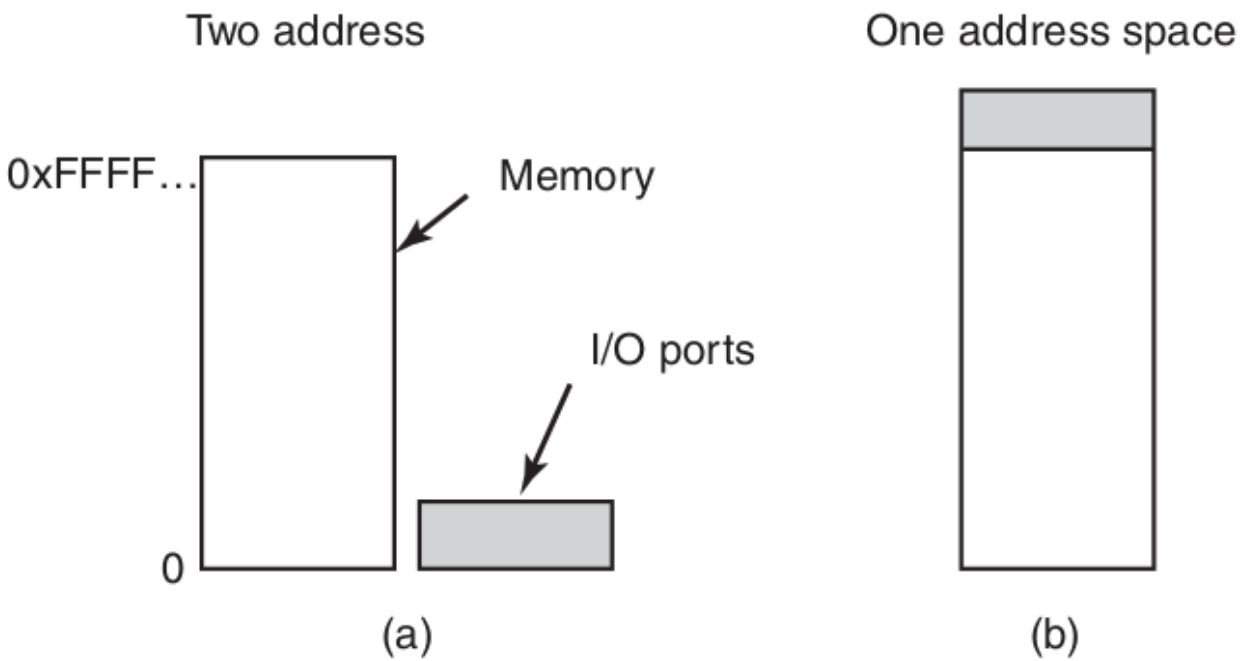
sú zariadenia, ktoré nemožno čítať ani zapisovať. Ich úlohou je reagovať na udalosti generovaním prerušení. (časovače, počítadlá, spínače...)

Štruktúra I/O zariadenia

- I/O Zariadenie môžeme rozdeliť na:
  - Samotné zariadenie.
  - Device Controller (Manažér zariadenia) označovaný aj ako adaptér.
- Device Controller môže obsluhovať:
  - práve jedno zariadenie. (Sériový port)
  - viacero zariadení rovnakého druhu. (SATA, SCSI)
  - viacero zariadení rôznych druhov. (USB, Thunder-bolt)
- Rozhranie Device Controller-a môžeme rozdeliť na:
  - riadiacu časť - riadiace a stavové registre.
  - dátovú časť - registre obsahujúce dáta a adresu prípadne data buffer.

Pripojenie I/O zariadenia k CPU

- Zariadenie ovládame a sledujeme prostredníctvom registrov.
- Niektoré zariadenia obsahujú aj malú internú pamäť.
- Existujú dva spôsoby ako zariadenie môžeme pripojiť k CPU:
  - Pripojenie na špeciálne porty CPU (Port I/O)
  - Pripojenie v adresnom priestore CPU (Memory-mapped I/O)



Pripojenie I/O zariadenia k CPU - Port I/O

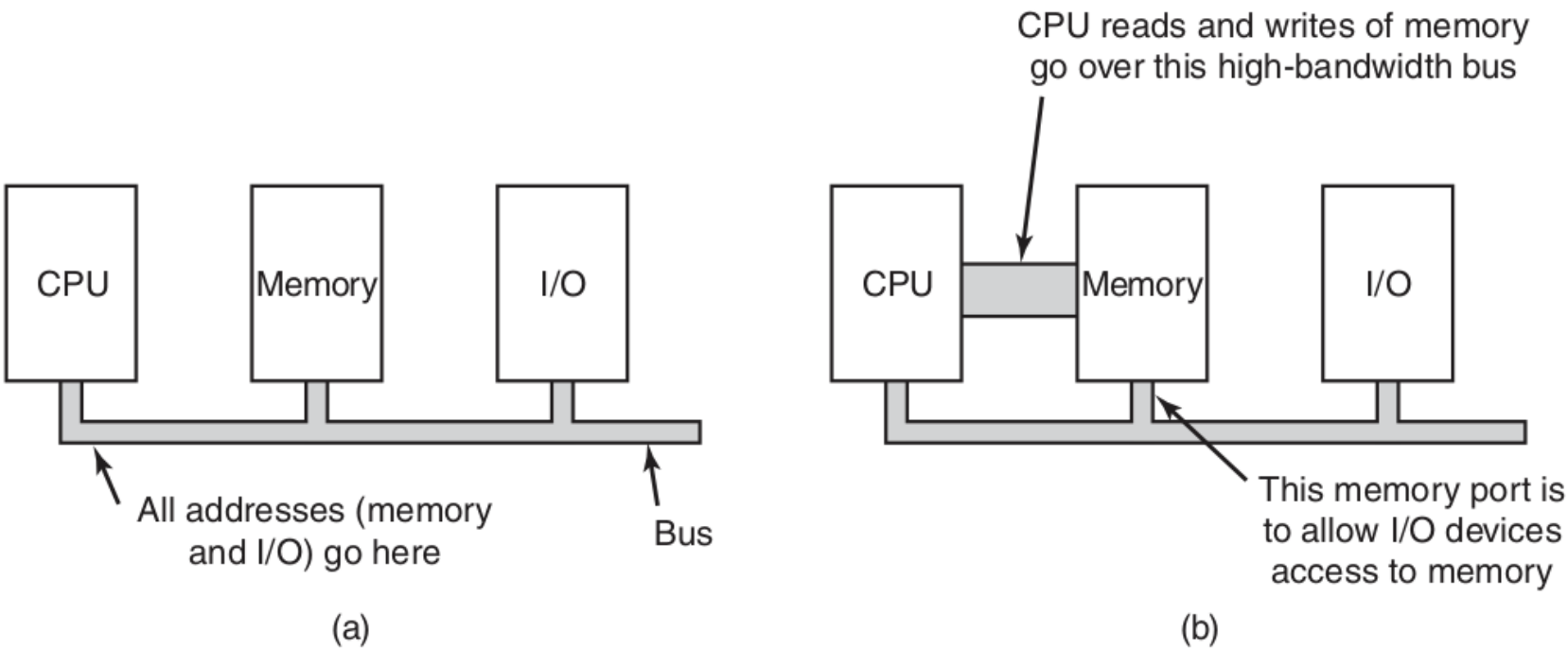
- Zariadenia sú pripojené k portom CPU v tzv. I/O port space.
- Prístup k zariadeniam prostredníctvom špeciálnych inštrukcií.
- Nevýhodou je, že špeciálne inštrukcie vyžadujú použitie Asembléru.
- OS je nutné vybaviť špeciálnymi procedúrami. Zväčšuje réžiu OS.

Pripojenie I/O zariadenia k CPU - Memory-mapped I/O

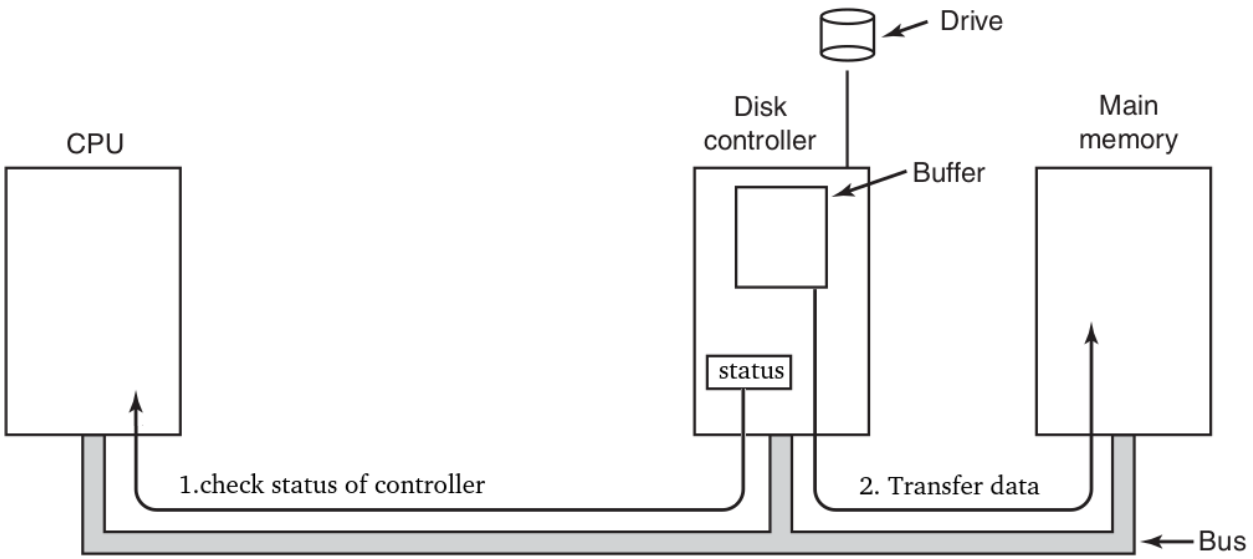
- Každý register zariadenia má svoju adresu v adresnom priestore procesora.
- Prístup k registru používa rovnaké inštrukcie ako prístup do pamäte.
- Z pohľadu používateľa je práca s I/O rovnaká ako práca s pamäťou.
- Zápis na konkrétnu adresu nevyžaduje špeciálnu procedúru.
- Prístup k zariadeniam sa dá riadiť jednoduchým obmedzením adresného priestoru procesu.
- HW CPU musí mať schopnosť zakázať cache nad zariadeniami.

Pripojenie I/O zariadenia k CPU - Memory-mapped I/O

- V minulosti bolo adresovanie zariadení jednoduché. Bola jedna zbernica. (a)
- Moderné CPU sú vybavené viacerými zbernicami a preto HW nutne musí rozlíšiť, či pristupujeme rýchlu pamäť alebo pomalšie zariadenie. (b)



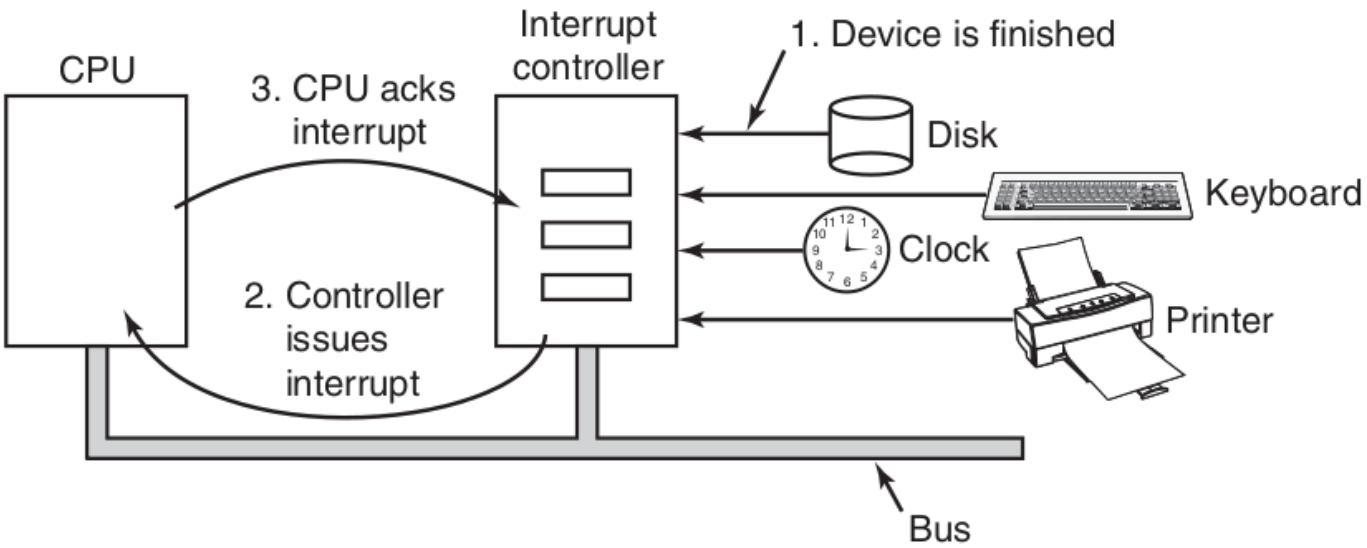
Polling zariadení



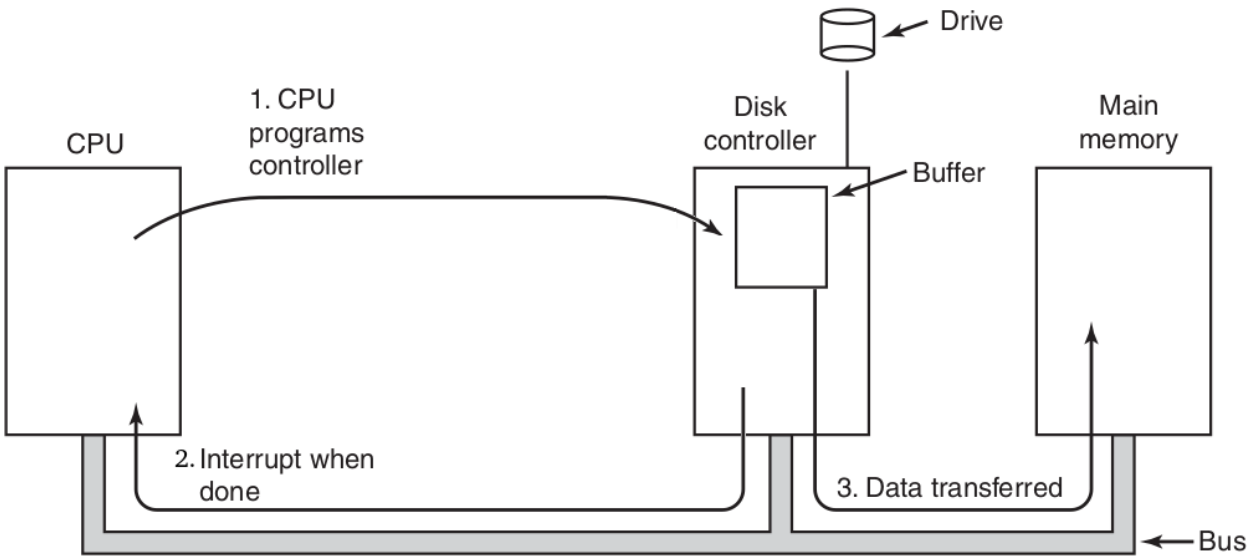
Prerušenia

Interrupt controller (IC)

je zariadenie, ktorého úlohou je zbierať všetky prerušenia, ktoré môžu v systéme nastať.



Prerušenia


Drive
1. CPU
programs
controller
Disk
controller
Main
memory
CPU
Buffer
2. Interrupt when
done
3. Data transferred
Bus

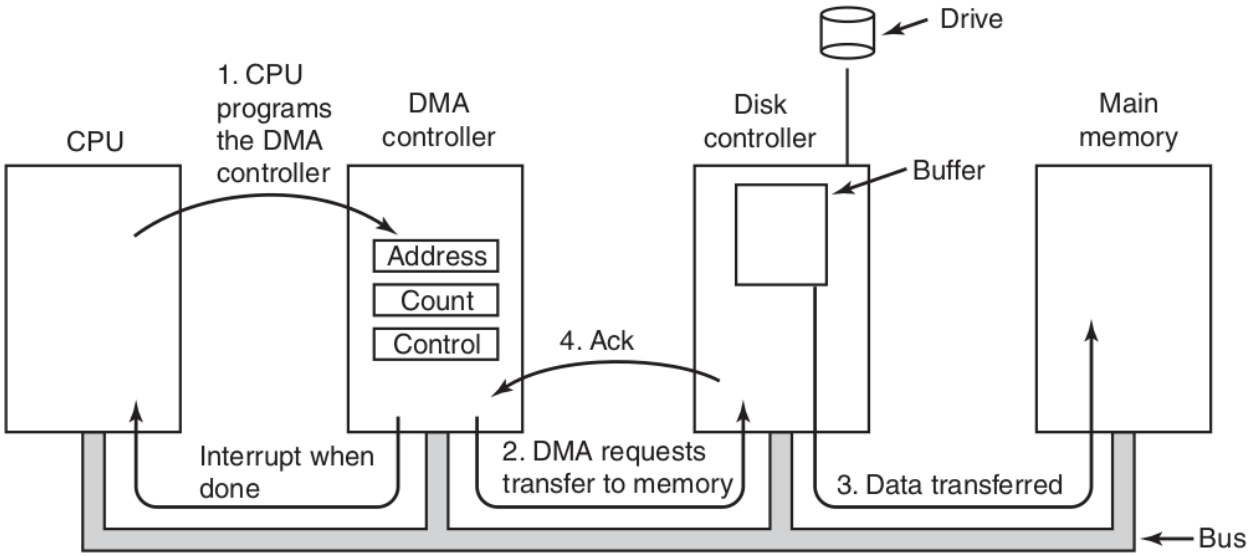
Prerušenia

- Každý zdroj prerušenia je registrovaný v IC.
- Prerušenia majú pridelené priority v systéme.
- Každé prerušenie je spracované IC nasledovne:
  1. Ak nastalo prerušenie, IC skontroluje, či neprebieha iné prerušenie.
  2. Ak nie, IC posunie na adresnú zbernicu číslo prerušenia a vyvolá prerušenie CPU.
  3. Ak áno, zariadenie, ktoré vyvolalo prerušenie čaká.
  4. Číslo prerušenia je odkazom na položku tabuľky prerušení.
  5. Položka, nazývaná Interrupt Vector (IV), obsahuje počiatočnú adresu pod-programu prerušenia.
  6. Pod-program prerušenia obsahuje kód, ktorý potvrdí spracovanie prerušenia zapísaním do registra IC
- Ak je CPU prerušené predtým než sa začne vykonávať pod-program prerušenia je nutné:
  - odložiť stav práve prebiehajúcej práce (Bežne je to stav procesu).
  - stav sa môže odložiť napríklad do zásobníka.

Direct Memory Access

DMA controller

je zariadenie, ktorého úlohou je vykonať prenos dát medzi zariadením a pamäťou bez nutnosti použitia CPU. Pri použití DMA, CPU nastaví adresu odkiaľ čítať dáta, adresu kam treba zapísať dáta a koľko dát chceme prečítať. DMA samé realizuje čítanie z adresy kým neprečíta všetky dáta. V takom prípade DMA vyvolá prerušenie CPU.



Ciele I/O SW

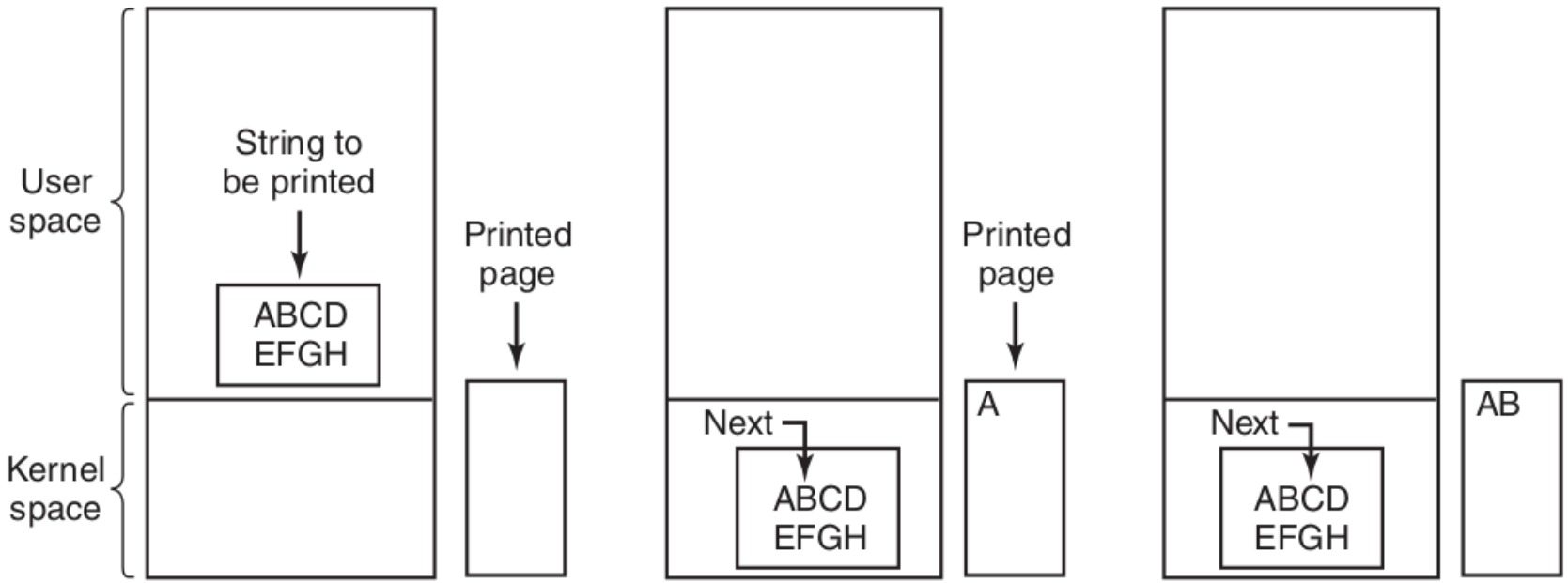
- Používateľský SW nezávislý od zariadení (transparentnosť).
  - Schopnosť písať programy, ktoré môžu používať akékoľvek zariadenie bez nutnosti identifikovania zariadenia.
  - Inými slovami ak čítam súbor musí byť jedno, že ho čítam z DVD alebo z disku.
- Uniformné pomenovanie zariadení.
  - Nezávislosť pomenovania zariadenia od jeho typu.
- Bottom-up error handling (náprava bez zásahu).
  - Chyby zariadenia by mali byť napravené v kontexte ovládača.
  - Ak dôjde k chybe, tak adaptér by sa mal pokúsiť ju napraviť.
  - Nasledovať by mal ovládač zariadenia. (Napr. opakovanie operácie)
- Synchrónne resp. blokujúce volania.
  - Blokujúce rozhranie je jednoduchšie z pohľadu používateľa.
  - Väčšina zariadení je asynchrónnych. OS musí takéto zariadenia zapúzdriť aby navonok boli synchrónne.
- Buffering
  - Dáta prijaté zo zariadení sú spracované ovládačom.
  - Používateľ už nemusí pracovať so zariadením. Stačí čítať z vyrovnávacej pamäte v RAM.

Implementácia I/O

- Možnosti ako implementovať ovládač I/O:
  - Programmed I/O - všetku prácu vykoná CPU
  - Interrupt-Driven I/O - CPU nastaví zariadenie a čaká na odpoveď
  - DMA-Driven I/O - CPU nastaví zariadenie a DMA a čaká na odpoveď

Programmed I/O

- Ovládač je jednoduchá procedúra.
- Vstupom procedúry je ukazovateľ na dáta od používateľa.
- Výstupom je výsledok operácie.
- Telo procedúry obsahuje slučku, ktorá číta stav zariadenia.



Programmed I/O

- Kód ovládača vyťažuje CPU, ktoré neustále kontroluje stav zariadenia.
- Cyklickému čítaniu stavu sa hovorí polling alebo busy waiting.
- Programmed I/O je výhodné len pri práci s rýchlymi znakovými zariadeniami.

```
1 copy_from_user(buffer, kBuffer, count);    // kBuffer is buffer in kernel space
2 for (i = 0; i < count; i++) {
3     while (*device_status_reg != READY);  // loop until device becomes ready
4     *device_data_reg = kBuffer[i];        // push character to device
5 }
6 return_to_user();
```

Interrupt-driven I/O

- Ovládač pozostáva z procedúry, ktorá inicializuje zariadenie, (a)
- a procedúry, ktorá sa zavolá počas vzniku prerušenia. (b)
- Proces, ktorý čaká na zariadenie môže byť preplánovaný a CPU môže vykonávať inú užitočnú činnosť.
- Interrupt-driven I/O je vhodné na pomalé znakové zariadenia

```
1 //(a) initialize device
2 copy_from_user(buffer, kBuffer, count);
3 enable_interrupts();
4 while (*device_status_reg != READY);
5 *device_data_reg = kBuffer[0];
6 block_user();
```

```
1 //(b) interrupt sub-rutine for device
2 if (count == 0) {
3     unblock_user();
4 } else {
5     *device_data_reg = kBuffer[i];
6     count--;
7     i++;
8 }
9 acknowledge_interrupt();
10 return_from_interrupt();
```

DMA-Driven I/O

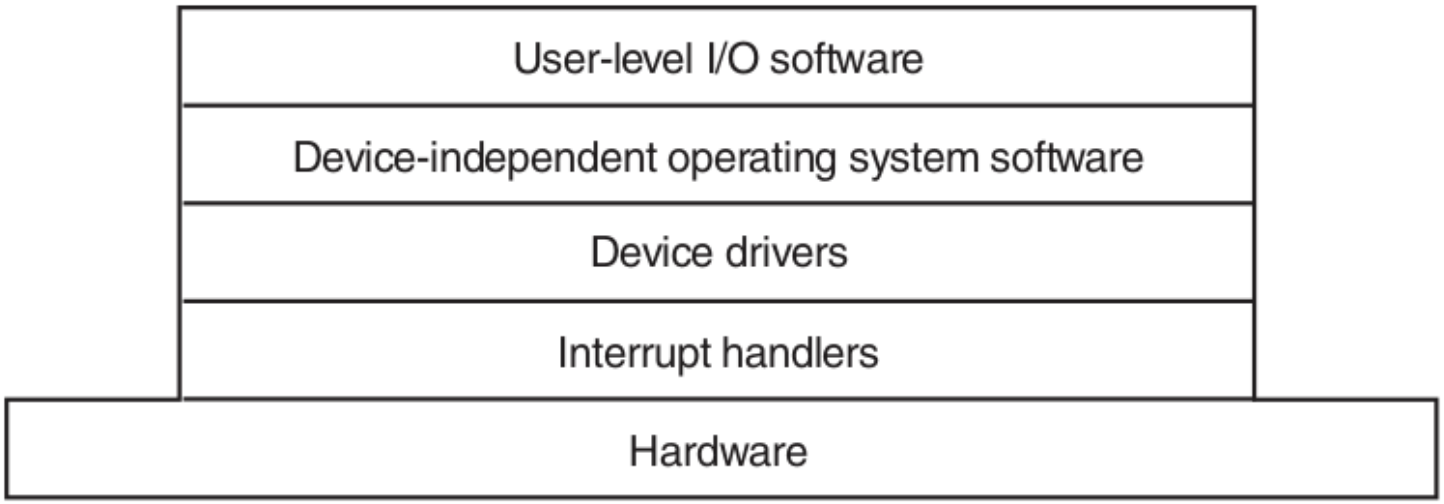
- Ovládač pozostáva z procedúry, ktorá inicializuje DMA. (a)
- Procedúry, ktorá sa zavolá po ukončení DMA. (b)
- Proces, ktorý čaká na zariadenie môže byť preplánovaný a CPU môže vykonávať inú užitočnú činnosť.
- DMA-driven I/O je vhodné na prácu s veľkým množstvom dát.
- DMA redukuje množstvo prerušení z jedného na znak na jedno na blok.

```
1 //(a) initialize DMA
2 copy_from_user(buffer, kBuffer, count);
3 set_up_DMA_controller(kBuffer, count);
4 block_user();
```

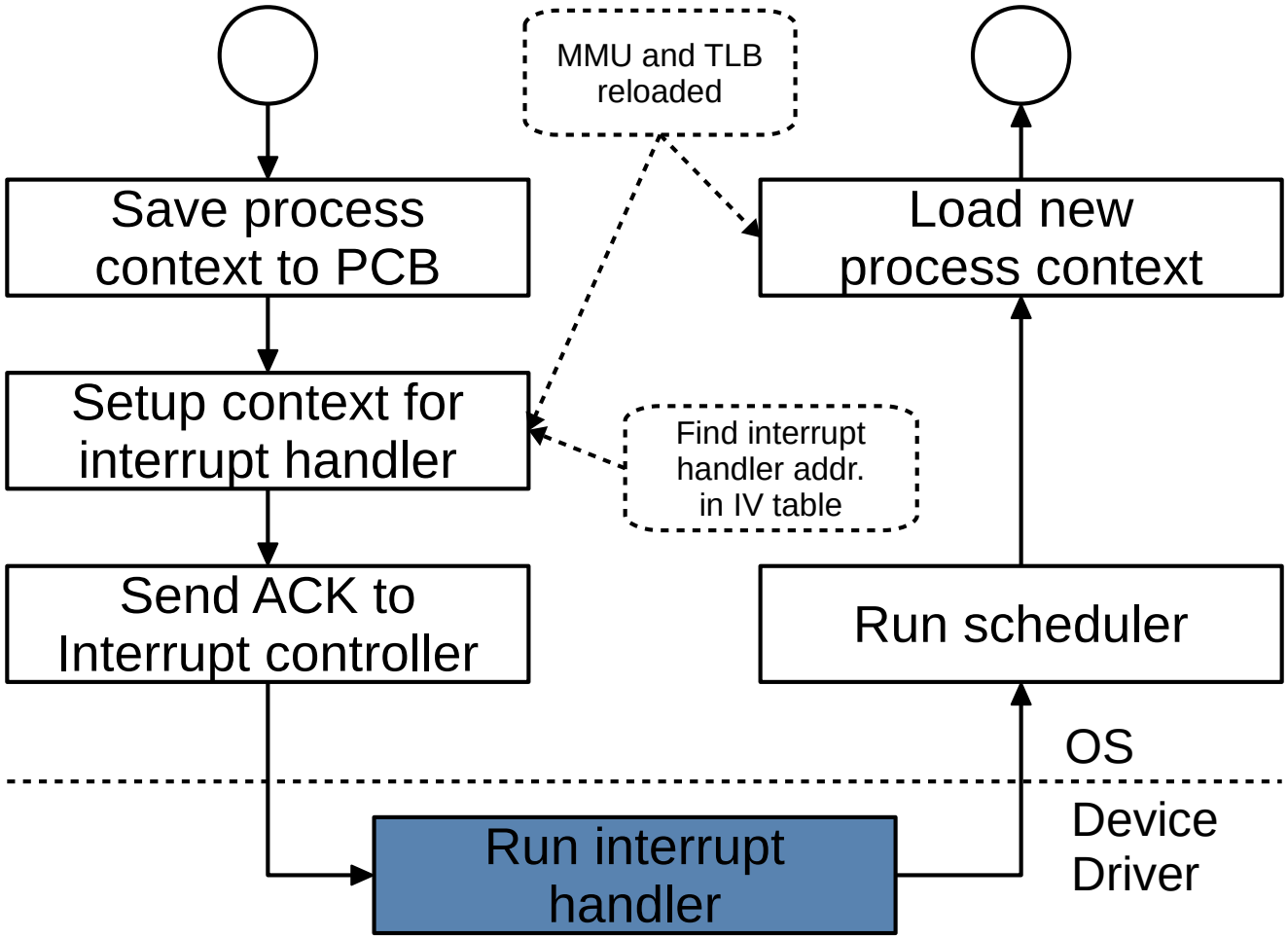
```
1 //(b) interrupt sub-rutine for DMA
2 unblock_user();
3 acknowledge_interrupt();
4 return_from_interrupt();
```

Vrstvy I/O SW

Vrstvy I/O SW


User-level I/O software
Device-independent operating system software
Device drivers
Interrupt handlers
Hardware

Interrupt Handlers



Device drivers

- Driver softvér, ktorý je napísaný zväčša výrobcom zariadenia.
- Device controller pozostáva z množstva riadiacich, stavových a dátových registrov.
- Úlohou driver-a je zapúzdriť tieto registre do vhodnejšieho rozhrania, ktorému bude rozumieť OS.
- Je potrebný ovládač pre každý OS.
- Identifikujeme viacero typov ovládačov:
  - Ovládač určený pre konkrétne zariadenia. (joystick, myš)
  - Ovládač určený pre špecifickú triedu zariadení. (SATA, SCSI)
  - Všeobecný ovládač (USB).

USB ovládače

Ovládače USB zariadení sú vrstvené podobným spôsobom ako TCP/IP stack. Najbližšie k HW býva link layer, ktorá dekóduje Byte stream na USB pakety. Vyššia úroveň sa zaoberá spracovaním paketov a inými spoločnými úlohami USB ovládačov. Najvyššia API úroveň sa zaoberá konkrétnym spacovaním požiadaviek na konkrétne USB zariadenia.

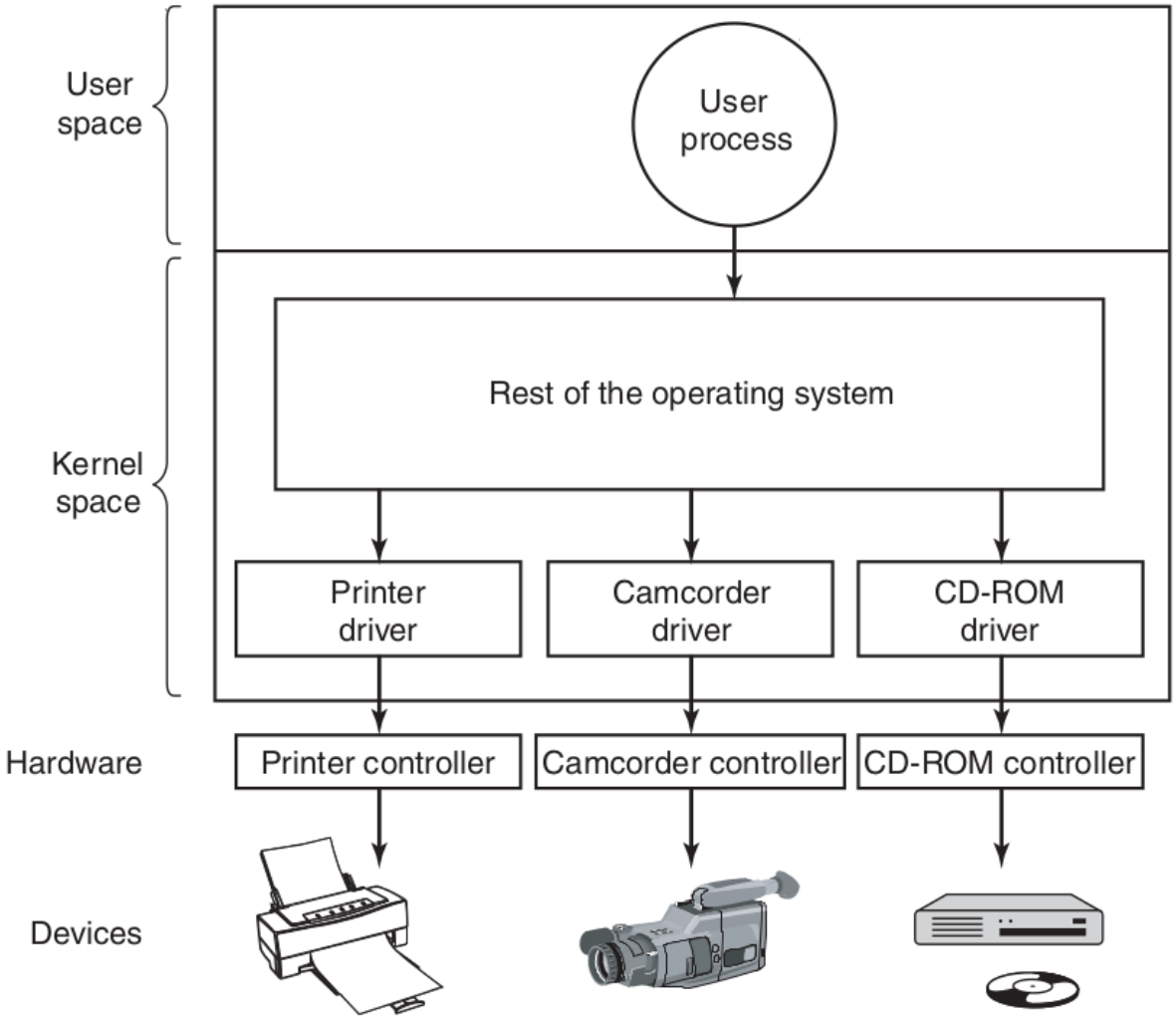
Device drivers

- Ovládače z princípu pristupujú k registrom, ktoré nemôžu byť prístupné používateľom.
- Preto ovládače prípadne ich časť musia byť súčasťou jadra OS.
- Návrhár OS je postavený pred dilemu. "Pretože do časti jeho ľubezného OS budú šahať cudzí ľudia."
- OS musí byť navrhnutý tak, aby bolo jednoduché OS rozšíriť o nové ovládače.
- Zároveň návrh musí zabezpečiť aby chybný ovládač nespôsobil paseku.

Device drivers

- V minulosti bolo nutné pridanie ovládača do zdrojového kódu OS a prekompilovanie OS.
- V súčasnosti OS bežne definujú štandardné rozhrania pre:
  - Block devices
  - Character devices
- Štandardné rozhranie pozostáva z množstva procedúr, ktoré ovládač musí implementovať.
  - init
  - read, write
  - power control
  - logging

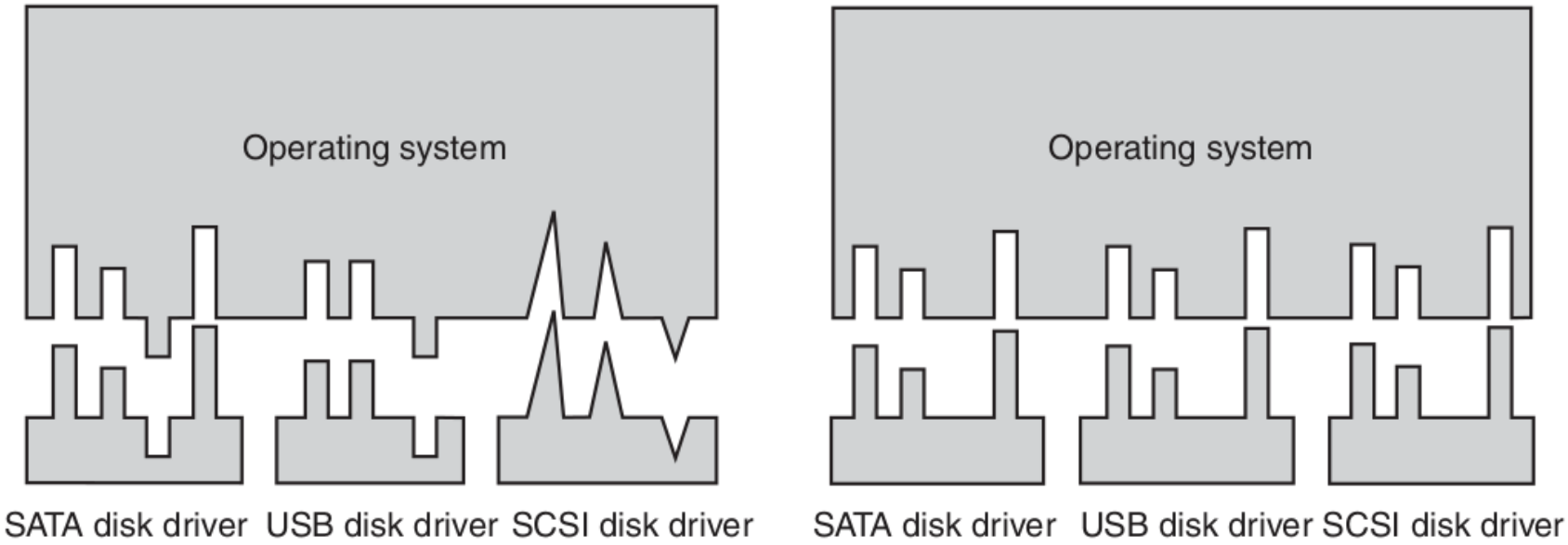
Device drivers



Device-Independent I/O SW

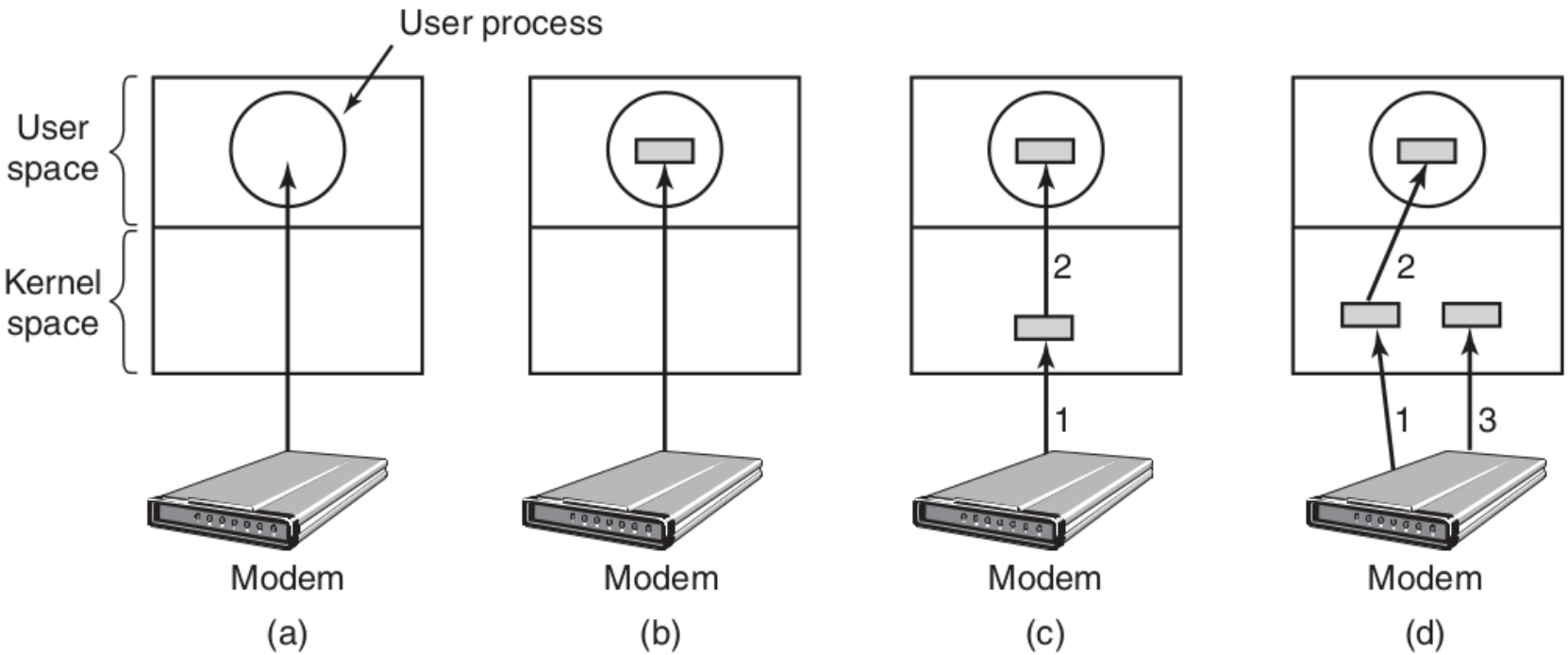
Uniform interfacing for device drivers
Buffering
Error reporting
Allocating and releasing dedicated devices
Providing a device-independent block size

Device-Independent I/O SW - uniformné rozhranie

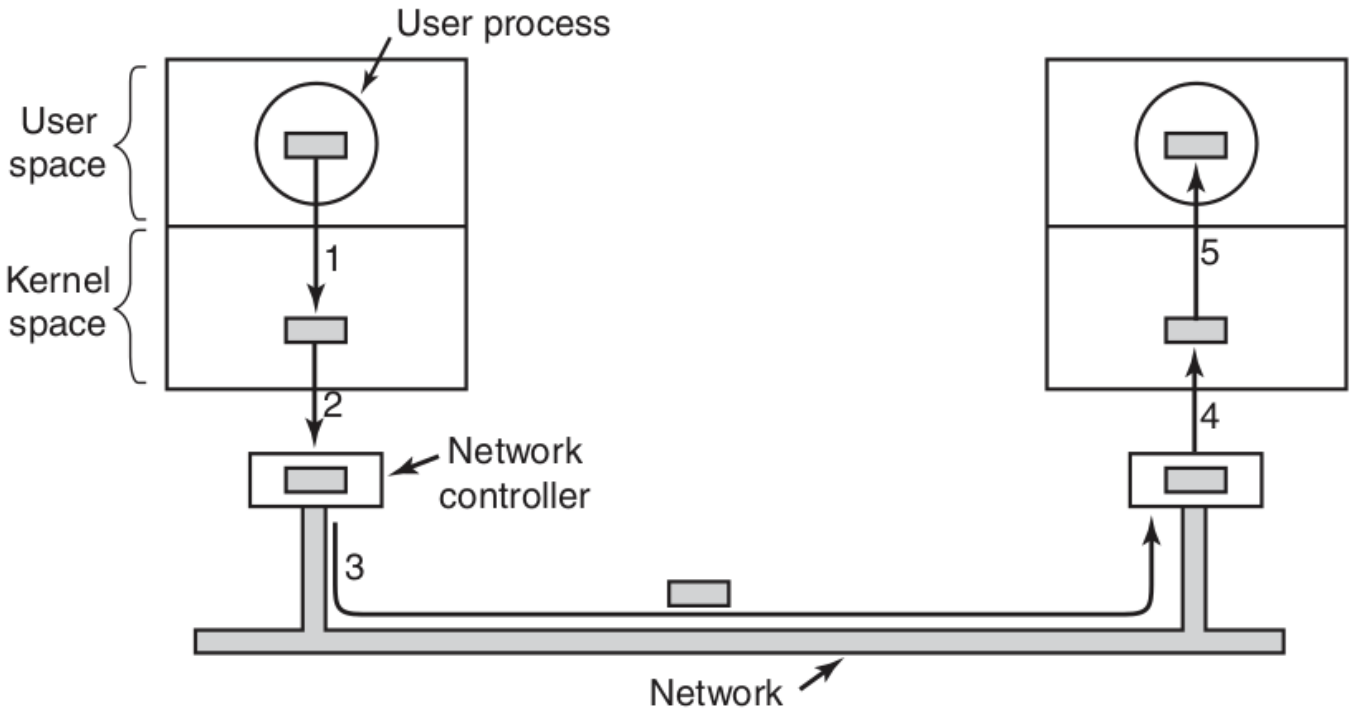


Device-Independent I/O SW - buffering

- Bez vyrovnávacej pamäte (a)
- Vyrovnávacia pamäť:
  - v procese (b)
  - v procese a jadre (c)
  - v procese a zdvojená v jadre (d)



Device-Independent I/O SW - buffering



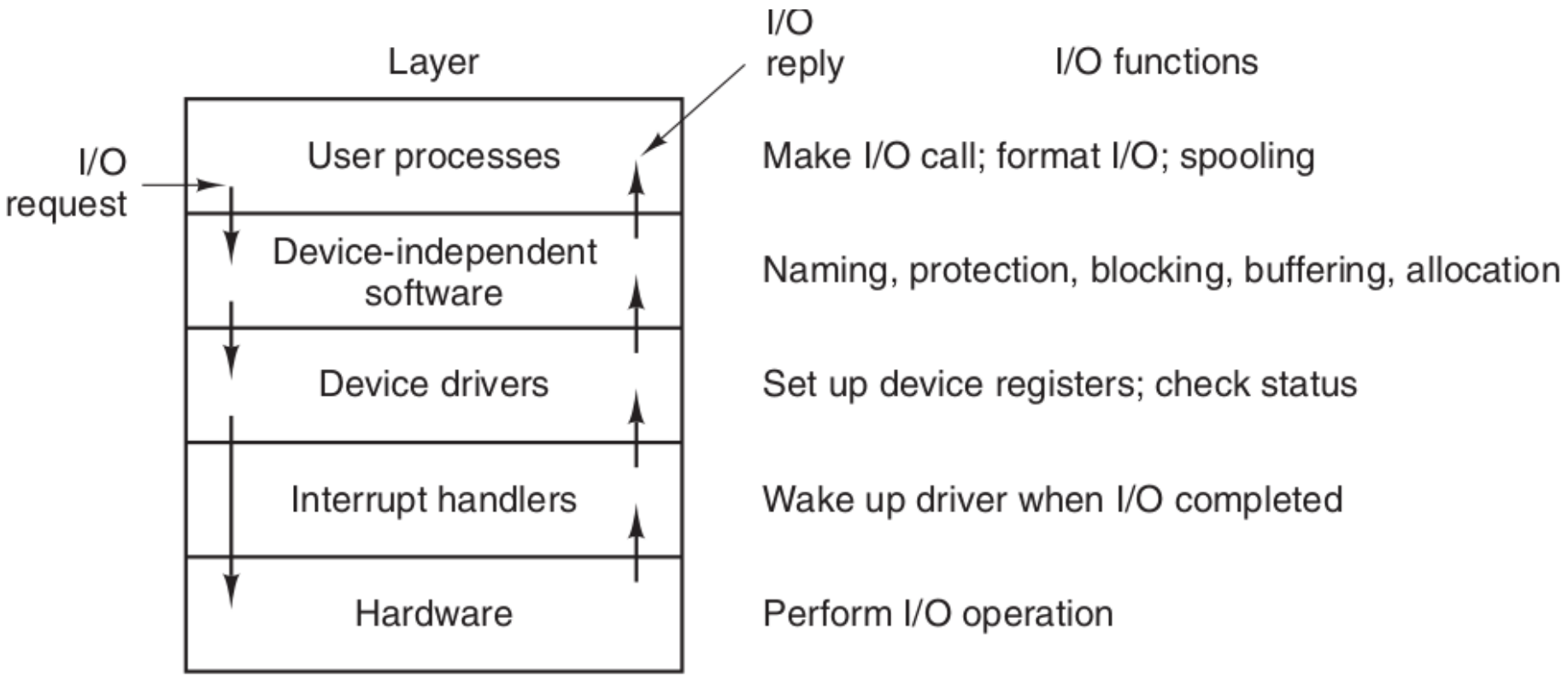
Device-Independent I/O SW

- Error reporting
  - Najbežnejším zdrojom chýb sú I/O zariadenia.
  - chyba programovania - ak proces chce nemožné. (Čítať z out zariadenia, prístup k neexistujúcemu zariadeniu.)
  - chyba I/O - zápis poškodeného bloku. (Čítanie z vypnutej kamery.)
- Riadenie prístupu k zariadeniu
  - Niektoré zariadenia vyžadujú single process access.
  - Aplikujú sa metódy vzájomného vylučovania.
- Nezávislá veľkosť bloku
  - Jednotlivé disky môžu mať rôznu veľkosť sektorov.
  - Navonok však OS pracuje len s jednou veľkosťou bloku.
  - OS a ovládač zabezpečuje správne adresovanie bloku na sektory.

User-space I/O SW

- Ovládač môže zabezpečiť všetku funkcionalitu zariadenia v jadre.
- Viac kódu v jadre OS znamená väčší potenciál pre chybné správanie výpočtového systému.
- Preto je bezpečnejšie mať väčšiu časť kódu ovládača v User-space.
- User-space časť ovládačov je reprezentovaná knižnicami.
- Bežné je okrem ovládačov zabezpečovať špeciálne procesy (daemony) riadiace I/O.

Vrstvy I/O SW


Layer
I/O reply
I/O functions
I/O request
User processes
Make I/O call; format I/O; spooling
Device-independent software
Naming, protection, blocking, buffering, allocation
Device drivers
Set up device registers; check status
Interrupt handlers
Wake up driver when I/O completed
Hardware
Perform I/O operation

Power Management

Power Management

- ENIAC 18,000 elektróniek -> 140kW.
- S príchodom tranzistorov klesla spotreba výpočtových systémov rapidne.
- Dlho sa spotreba neriešila. Ale posledné roky sa spotreba znovu stáva témou.
  - Prenosné zariadenia.
  - Dátové centrá.
  - IOT a EKO generation.

Power Management

Prečo riešiť spotrebu?

V súčasnosti Bežné desktop PC má spotrebu 100-200W. Predstavme si, že takýchto PC naraz pustíme 100 miliónov. Ak to všetko spočítame tak ich celková spotreba je 20 GW. To je zhruba 20 priemerných jadrových elektrární. Ak sa nám podarí znížiť spotrebu počítača na polovicu môžeme si dovoliť vypnúť 10 takýchto elektrární.

Prečo riešiť spotrebu II?

Napriek obrovským investíciám do výskumu batériových zdrojov, je nárast kapacity batérií veľmi malý. Ak chceme zachovať Moorov zákon je ich spotreba v hľadáčiku každého výrobcu čipov.

Power Management - OS

- Najväčšími spotrebiteľmi energie sú: display, CPU, HDD.
- Návrh OS tiež môže prispieť k zníženiu spotreby:
  - Zníženie výkonu a frekvencie CPU ak nie je zaťažené.
  - Ovládanie jasu obrazu pri neaktivite PC.
    - LCD - jas(backlight) sa môže podieľať na 10-15% spotreby laptop-u.
    - OLED - vplyv na spotrebu má výber pozadia.
  - Uspanie neaktívnych zariadení. Vypnutie rotácie disku ak sa nepoužíva.

Zhrnutie

Zhrnutie

- I/O SW je špecifický nakoľko jeho časti musia byť zakomponované do jadra OS.
- Avšak tento SW nepíšu návrhári OS ale výrobcovia zariadení.
- Je na návrhárovi OS aby poskytol čo najvšeobecnejšie a najjednoduchšie rozhranie OS-ovládač.
- I/O SW možno rozdeliť na:
  - platformovo závislú časť: Interrupt handler a zapúzdrujúci kód nad HW.
  - platformovo nezávislú časť: riadenie prístupu, manažment, buffering...

Čo robiť do ďalšej prednášky

- Prečítať kapitolu 10. a 11. z Tanenbauma