Operačné systémy

2. Úvod do OS

Ing. Martin Vojtko, PhD.

2023/2024

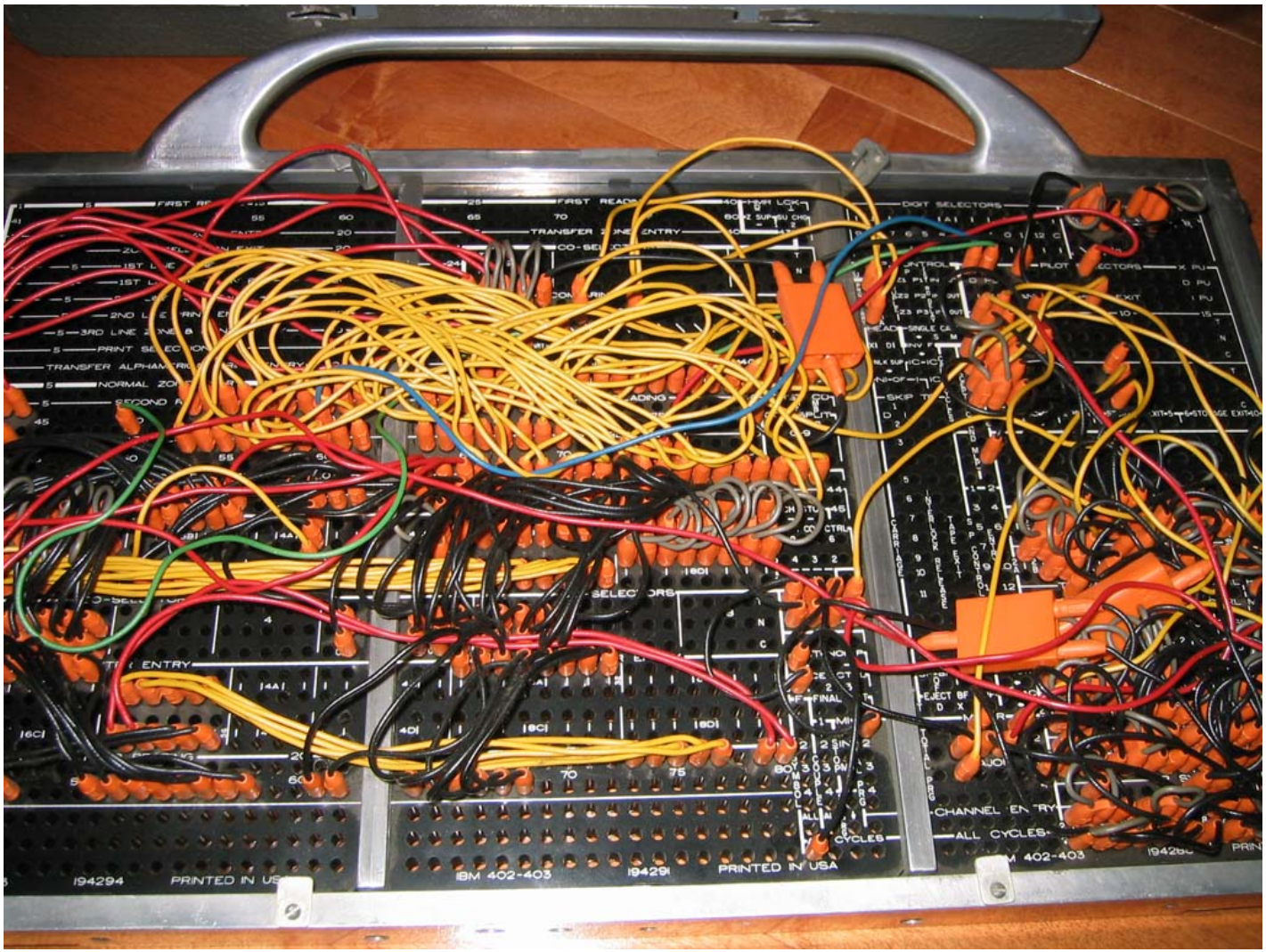
1. História OS
2. Klasifikácia OS
3. Stručný opis HW
   - Procesor - CPU
   - Pamäť
   - I/O zariadenia
4. Stavebné kamene OS
   - Procesy
   - Adresný priestor
   - Súborové systémy
   - Používateľské rozhrania
   - Systémové volania
5. Zhrnutie

História OS

1. generácia (1940-1955)

- **Plugboard** - Používajú sa mechanické relé, vákuové trubice (elektrónky) a programovacie dosky.
- Nedá sa hovoriť o existencii OS v pravom slova zmysle.
- Programovanie priamo na doske plošných spojov a v strojovom kóde.
  "Keď zapojím ten kábel sem tak by to... môžem ísť po novú elektrónku."
- Výpočtový prostriedok použitý výlučne na riešenie matematických problémov.
  "Pane myslím, že sme prelomili Enigmu. Ešte sa treba naučiť Nemecky."
- Personál musí navrhovať/programovať/vykonávať/poznať všetko.
- Neexistuje ani len Asemblér.

1. generácia (1940-1955)



1. generácia (1940-1955)

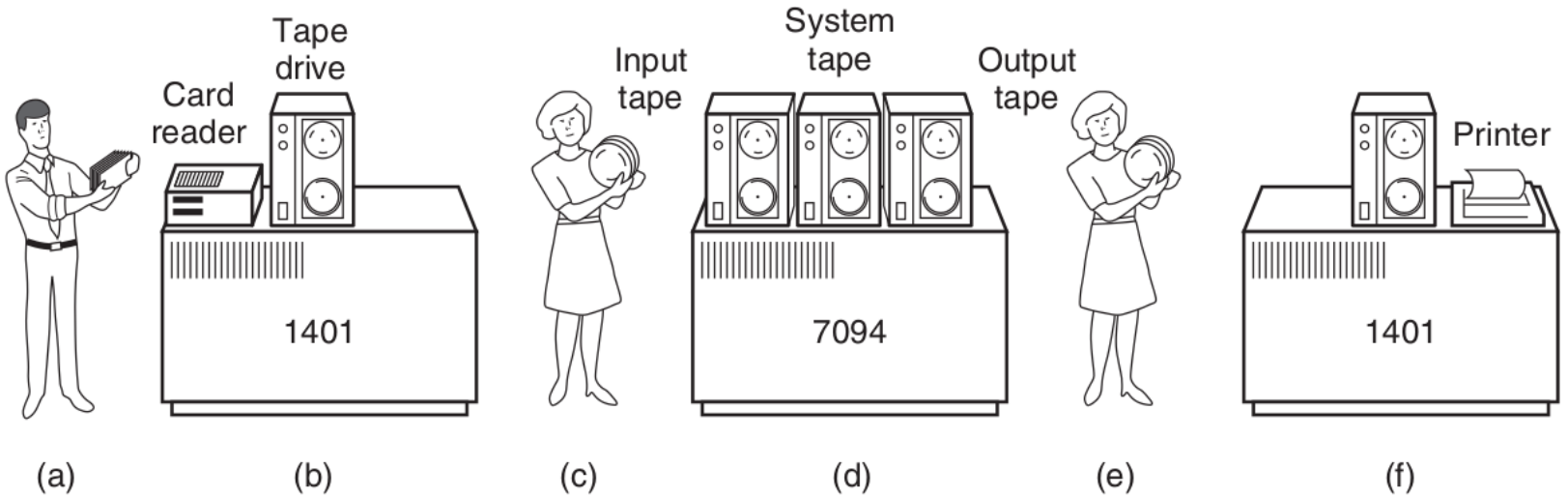
Ako to bolo vtedy?

Prakticky každý počítač je postavený na mieru a vyžaduje veľmi drahé školenie personálu od výrobcu. Je časté, že personál je súčasťou balenia. Ak už ste zaškolený pravdepodobne vás neprepustia. Porucha počítača je na dennom poriadku. Takže každý výpočet sa pravdepodobne musí opakovať niekoľko krát.

2. generácia (1955-1965)

- **Tranzistor** (1950) - Výpočtové systémy sú spoľahlivejšie. Šíria sa aj do korporácií.
- Personál sa začína špecializovať na vývojárov, operátorov, programátorov a obslužný personál.
- **Mainframe** - Najbežnejšia forma výpočtových systémov. Mainframe rad za radom vykonáva úlohy (**Jobs**) v dávkach (**Batches**).
- Jednotlivý výrobcovia (predovšetkým IBM) vyrábajú počítače v produktových radách. Snažia sa štandardizovať aby minimalizovali náklady na výrobu.

2. generácia (1955-1965)



2. generácia (1955-1965)

2. generácia (1955-1965)

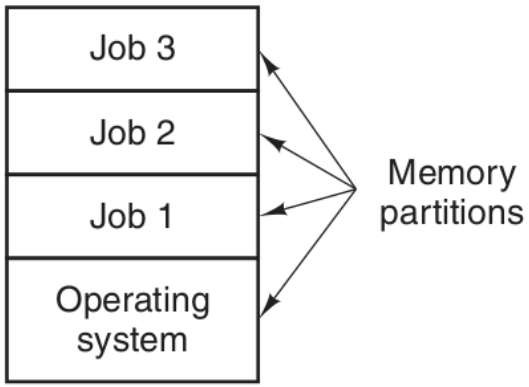
Ako to bolo vtedy?

Na programovanie sa používa Asemblér a Fortran. Program sa napíše na papier a následne sa prepisuje dierovaním na dierne štítky. Programy (Jobs) sa odovzdajú operátorovi, ktorý ich postupne nahraje na magnetickú pásku v tzv. Input room. Vytvorí tzv. Batch (dávka). Páska sa vloží do mainframe-u. Okrem Batch pásky sa do systému nahrá program, ktorého úlohou je načítať prvý program z Batch pásky. Tento program môžeme označiť za predchodcu OS. Všetci si dajú kávičku a čakajú na výsledky. Po spracovaní dávky operátor presunie pásku s výstupmi do tzv. Output room kde sa vytlačí jej obsah. Programátor tradične ide opraviť chybu v programe. Čas, výkon a pamäť sú vzácne preto je snahou mať optimálne programy.

3. generácia (1965-1980)

- **Integrated Circuits (ICs)** - Tranzistory sa zmenšujú a združujú do integrovaných obvodov.
- Štandardizácia vedie k vzniku produktových rodín. Konkrétne rodiny sú vybavené rovnakým softvérom OS.
- **Multiprogramming** - Počítač udržiava v pamäti viac úloh. Bežne existuje niekoľko slotov pamäte rovnakej veľkosti. Ak nejaká úloha čaká na I/O tak iná úloha sa vykonáva.
- **Spooling** - Počítač zaraďuje nové úlohy do hlavnej pamäte akonáhle sa nejaký slot uvoľní. I/O počítače sú minulosťou.
- **Batch systems** - Dovtedajšie systémy spracovali úlohy dávkovo. Odovzdanie spracovanie a vyhodnotenie úlohy môže trvať niekoľko hodín.

3. generácia (1965-1980)



3. generácia (1965-1980)

- **Timesharing** - variant Multiprogramming-u, kde sa striedajú používatelia namiesto úloh.
  - Súčasne môže pracovať a zdieľať systém mnoho používateľov. "Aj tak 2/3 používateľov pije kávičku."
  - Obrat vykonania úlohy sa skracuje z hodín na minúty.
- **Terminals** - Terminál je spôsob ako používateľ zadáva príkazy mainframe-u. Implementácia Timesharing-u.
- **Minicomputers** - "menší" počítač určený na spracovanie jednoduchších úloh. Ako konkurencia k mainframe-om a terminálom
- OS/360 neskôr MULTICS -> UNIX -> chaos -> POSIX -> MINIX -> Linux

3. generácia (1965-1980)

Ako to bolo vtedy?

Bežný OS (OS/360) je neprehľadný kus softvéru. Veľká časť je implementovaná v Asembléri. Milióny riadkov kódu spravovaných batalión om programátorov. Oprava jednej chyby vedie ku vzniku nových. Z nejakých záhadných príčin si to zákazníci nevšimli.

Ako to bolo vtedy?

Vtedajšia predstava sveta bola, že jeden centrálny počítač obslúži tisícky používateľov. Používateľovi mal stačiť jednoduchý terminál. Nepripomína vám to Cloud?

3. generácia (1965-1980)

Ako to bolo vtedy?

Ken Thompson našiel jeden nestrážený Minicomputer PDP-7. Za dlhých zimných večerov prepísal MULTIX na OS pre jedného používateľa, UNIX. Každému sa to páčilo a preto UNIX softvér prevzali a upravili si ho pre svoje potreby. Istá organizácia (IEEE) si následne povedala že im do toho bude hovoriť a vznikol štandard POSIX.

4. generácia (1980-)

- **Large Scale Integration (LSI)** - Jednotlivé čipy obsahujú tisícky tranzistorov.
- **Personal Computer (PC)** - Nástup osobných počítačov. Sú cenovo dostupné pre jednotlivcov.
- Búrlivý vývoj OS. Vzniká množstvo druhov OS rôznej špecializácie.
- Rozšírenie grafických rozhraní umožňuje rozšírenie PC do domácností.
- Rozvoj počítačových sietí.

4. generácia (1980-)

Ako to bolo vtedy?

Z pohľadu dneška došlo k mnohým mnohým zlým rozhodnutiam manažmentu niektorých firiem. Intel prišiel o šancu ovládnuť trh s operačnými systémami. Bill Gates kúpil DOS od Tima Patersona za 75 000$. Tim sa neskôr stal zamestnancom MicroSoft-u. DOS neskôr MS-DOS pohltí celý trh s IBM počítačmi. Steve Jobs zbadá potenciál v bezvýznamnom projekte spoločnosti Xerox hrali sa tam s oknami. Jobs uviedol na trh používateľsky prívetivý počítač s GUI. Počítač zameraný na klientelu, ktorá "nič nevie a ani nechce vedieť".

5. generácia (1990-)

- **Very Large Scale Integration (VLSI)** - Jednotlivé čipy obsahujú milióny tranzistorov.
- **Mobile Computer, Smartphone** - Nástup mobilných zariadení, ktoré sa nepoužívajú len na telefonovanie.
- Rozvoj bezdrôtových sietí. S nimi príchod špecializovaných OS pre mobilné zariadenia.

5. generácia (1990-)

Ako to bolo vtedy?

Prvý smartphone, ktorý kombinuje telefón a PDA uviedla NOKIA. Prvým operačným systémom používaným v týchto zariadeniach bol Symbian. Neskôr pribudli RIM (Blackberry 2002) a IOS (Apple 2007). Nakoniec ale prevládol Android. V 2011 sa NOKIA rozhodla, že prejde na Windows čím pravdepodobne nadobro potopila Symbian. Android je open-source čo je veľmi lákavé pre výrobcov telefónov. Rovnako má silnú základňu developerov.

Klasifikácia OS

Klasifikácia podľa určenia

- Mainframe OS
- Server OS (Multiprocessor OS)
- PC OS
- Embedded OS (Real-time OS)

Klasifikácia podľa určenia - Mainframe OS

Mainframe OS

Je OS určený predovšetkým na spracovanie veľkého počtu jednoduchých úloh, Job-ov. Mainframe systémy sú vyladené a stabilné systémy. Sú schopné pracovať celé roky bez odstávky. Typicky sa používajú na Batch a transakčné spracovania a Timesharing. Batch spracovanie sa používa pri zbieraní dát z reťazcov pobočiek. Transakčné spracovanie sa používa pri spracovaní jednoduchých operácií aké sú bežné napríklad v bankách. Timesharing je bežne používaný pre zabezpečenie prístupu veľkého počtu používateľov do veľkých databáz.

Klasifikácia podľa určenia - Server a PC OS

Server OS

Je OS určený predovšetkým pre zabezpečenie prístupu veľkého počtu používateľov. Bežnými úlohami Server OS je zdieľanie HW a SW zdrojov. Server ako HW sa vyznačuje širokou škálou výkonu od silnejších PC po stroje dosahujúce výkon Mainframe-ov.

PC OS

Bežný OS, ktorý nájdeme v PC. Zväčša určený pre jedného používateľa vykonáva úlohy bežnej potreby. V súčasnosti takýto OS beží na mnohojadrových procesoroch. Rozdiely medzi Server OS a PC OS sa prakticky zužujú na špecifické nastavenia jadra operačného systému.

Klasifikácia podľa určenia - Multiprocessor OS

Multiprocessor OS

Je OS určený predovšetkým pre výpočtové systémy, ktoré sú vybavené množstvom procesorov združených do tzv. Clusters. Nasadzujú sa na výpočtovo náročné úlohy ako je počítanie matematických problémov prípadne predpovedí počasia.

Klasifikácia podľa určenia - Embedded a Real-time OS

Embedded OS

je určený na riadenie výpočtových systémov, bez priameho zásahu používateľa. Nasadzuje sa do menších prístrojov dennej potreby ako je práčka, sušička. Na druhú stranu existujú vnorené systémy, ktoré riadia výrobný proces ako je napríklad v pivovare, vysokých peciach.

Real-time OS

svojimi vlastnosťami pripomína Embedded OS. Dôležitým rozdielom je stanovená doba odozvy, ktorá musí byť za každú cenu dodržaná. Podľa kritickosti nedodržania doby odozvy rozlišujeme Soft RT a Hard RT OS.

Klasifikácia podľa štruktúry

Monolitické

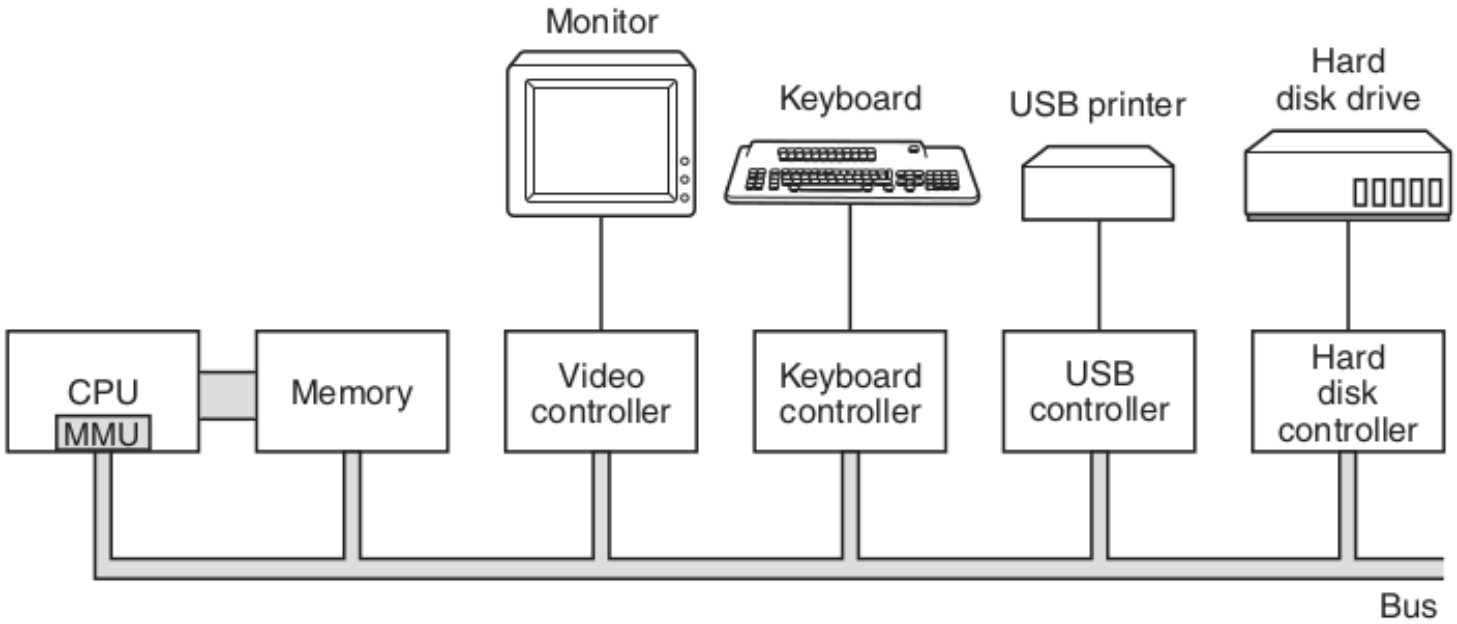
Jadro OS je jeden veľký blok bežiaci v privilegovanom režime. Jedna funkcia môže volať druhú bez reštrikcií. Tisíce funkcií otvárajú veľký priestor na chybu, ktorá často krát vedie k zrúteniu systému. Na druhú stranu bez rozhraní a pravidiel je možné dosiahnuť nízku latenciu systému.

Microkernel

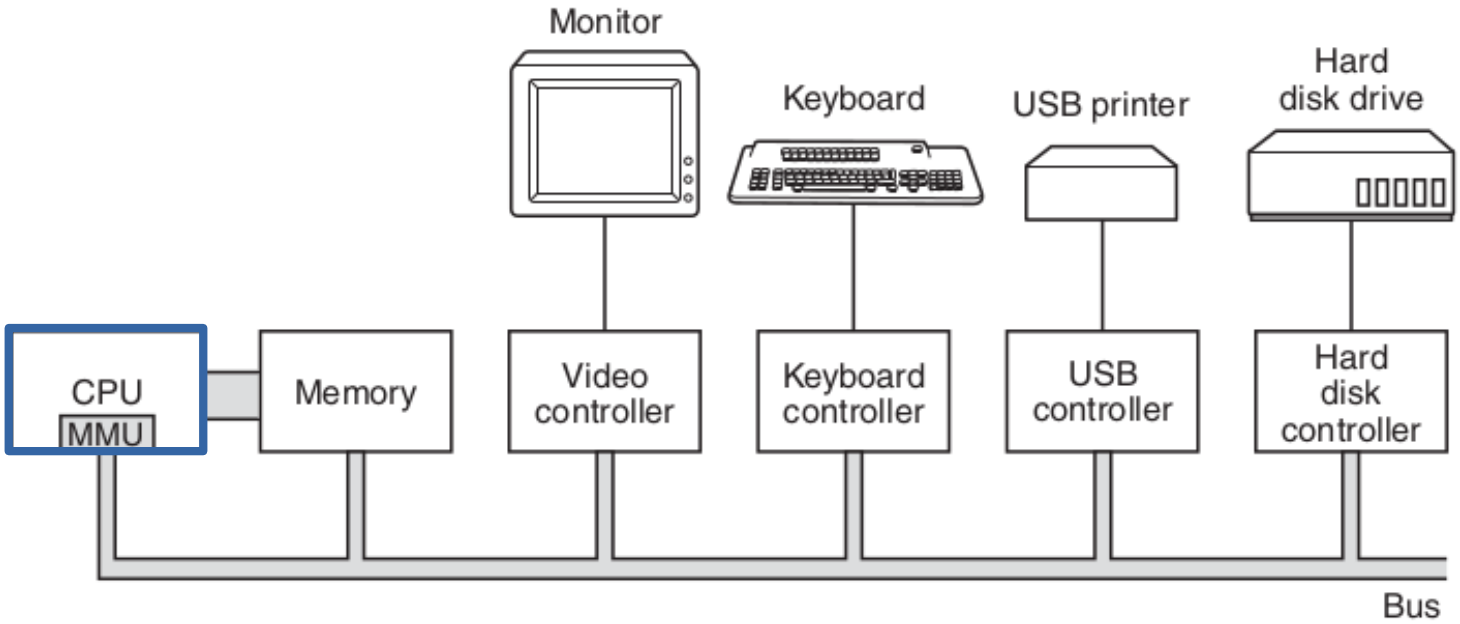
Jadro OS obsahuje minimum potrebné na abstrakciu HW (správa pamäte a I/O, obsluha prerušení). Toto jadro beží v privilegovanom režime. Ostatné časti OS sú vykonávané ako služby/procesy v používateľskom režime. Medzi jednotlivými službami OS je nutná IPC. OS je stabilný lebo zrútenie jednej služby nevedie k zrúteniu celého OS. Jednotlivé volania OS majú vyššiu latenciu.

Stručný opis HW

Stručný opis HW



Procesor - CPU



Procesor - CPU

- Mozog výpočtového systému. Vykonáva program inštrukciu za inštrukciou.
- Rozhraním pre OS sú:
  - Množina inštrukcií
  - Registre (všeobecné, PC, SP, PSW, ...)
  - Režimy CPU (kernel, user ...)
- Situáciu komplikuje architektúra CPU
  - pipeline
  - multithreading
  - multicore
  - prerušovací podsystém

Procesor - CPU

Režim CPU

určuje akú podmnožinu inštrukcií je možné vykonávať v programe. Väčšina CISC CPU má niekoľko takýchto režimov. Kernel OS je vykonávaný obyčajne v najviac privilegovanom režime. Prechod medzi menej a viac privilegovaným režimom je možný len pomocou špeciálnej inštrukcie (TRAP). Táto inštrukcia je súčasťou systémového volania OS.

Procesor - CPU

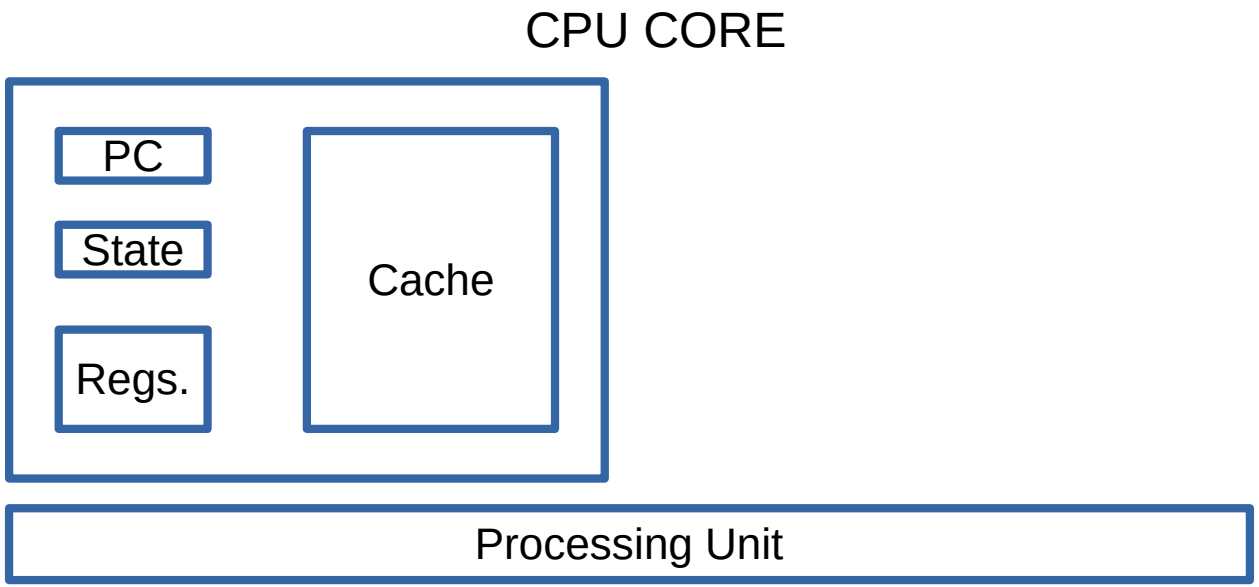
Multithreading - Hyperthreading

Moderné procesory podporujú uloženie kontextov dvoch vlákien naraz. Vždy sa vykonáva práve jedno vlákno. Výhoda je, že v prípade ak jedno vlákno je blokované, tak v jednotkách ns je možné prepnúť kontext spracovania na druhé vlákno. Je to výrazná úspora času pretože bežné prepnutie môže trvať rádovo dlhšie.

Procesor - CPU

Multithreading - Hyperthreading

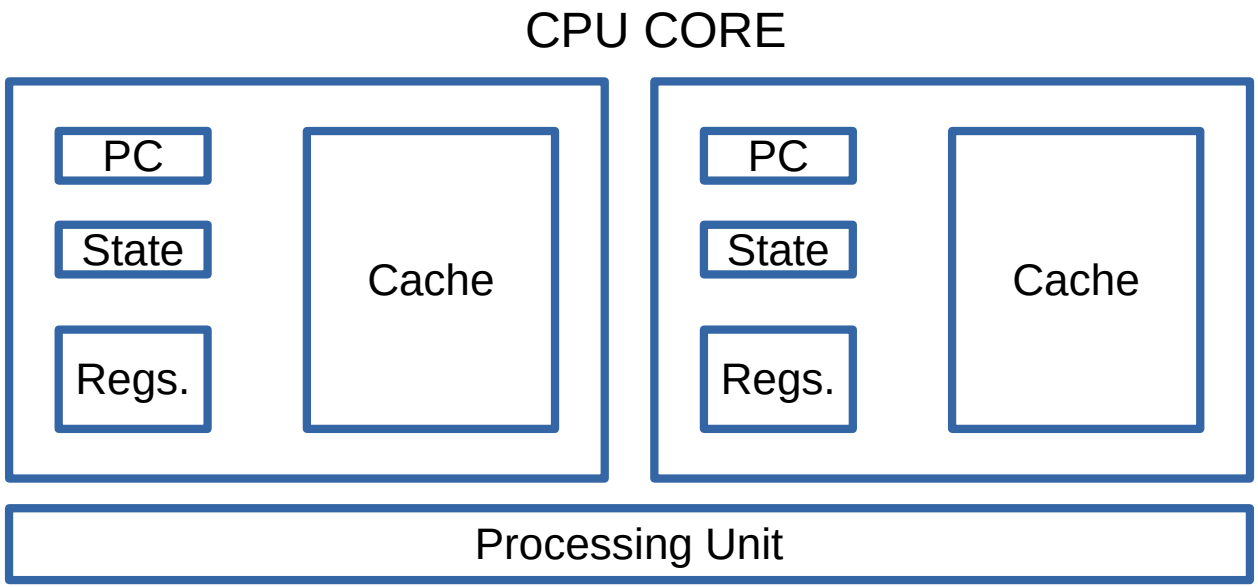
Moderné procesory podporujú uloženie kontextov dvoch vlákien naraz. Vždy sa vykonáva práve jedno vlákno. Výhoda je, že v prípade ak jedno vlákno je blokované, tak v jednotkách ns je možné prepnúť kontext spracovania na druhé vlákno. Je to výrazná úspora času pretože bežné prepnutie môže trvať rádovo dlhšie.



Procesor - CPU

Multithreading - Hyperthreading

Moderné procesory podporujú uloženie kontextov dvoch vlákien naraz. Vždy sa vykonáva práve jedno vlákno. Výhoda je, že v prípade ak jedno vlákno je blokované, tak v jednotkách ns je možné prepnúť kontext spracovania na druhé vlákno. Je to výrazná úspora času pretože bežné prepnutie môže trvať rádovo dlhšie.



Procesor - CPU

Prerušovací podsystém

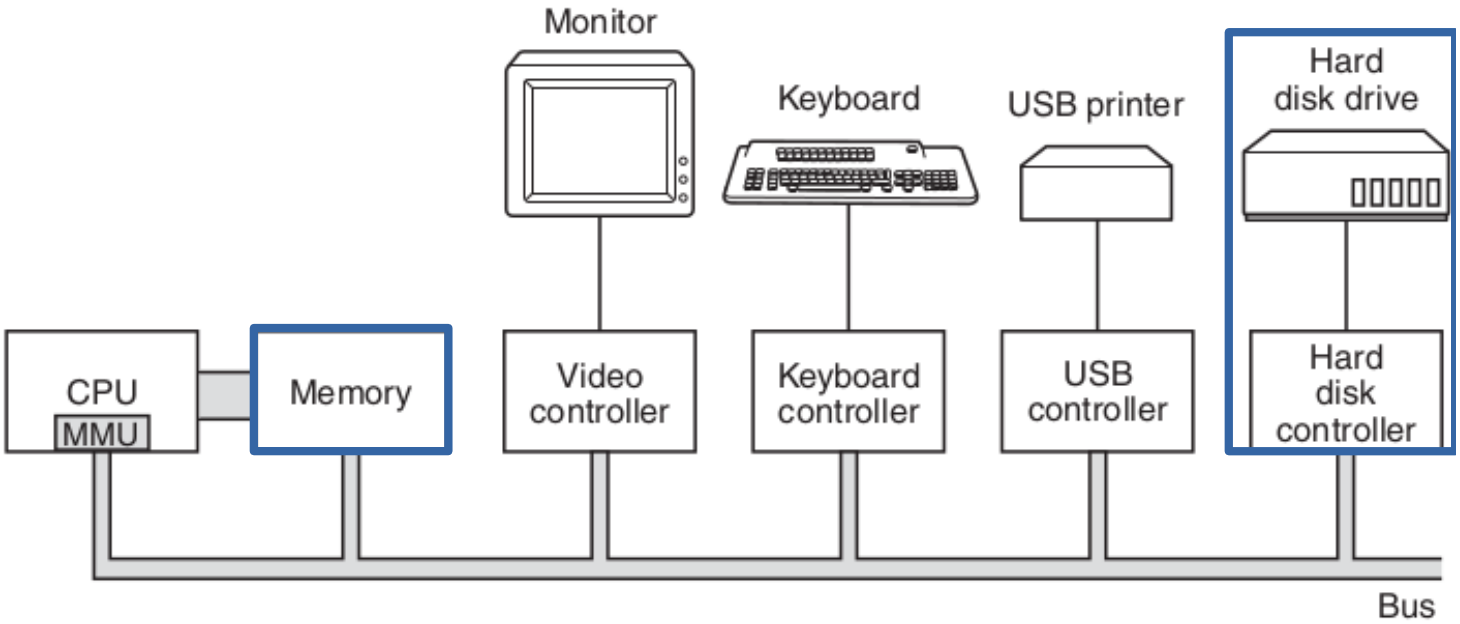
Procesor vykonáva prácu sekvenčne. Na to aby bolo možné vykonávať prácu viacerých úloh bol do CPU zavedený mechanizmus prerušenia práce. Prerušenie CPU spôsobí, že register PC je prepísaný novou adresou, z ktorej je prečítaná prvá inštrukcia. Obyčajne na tejto adrese je uložený prerušovací podprogram OS, ktorý zabezpečí správne odloženie kontextu prerušenej úlohy a následné spracovanie prerušenia.

Procesor - CPU

Čo musí OS zabezpečiť

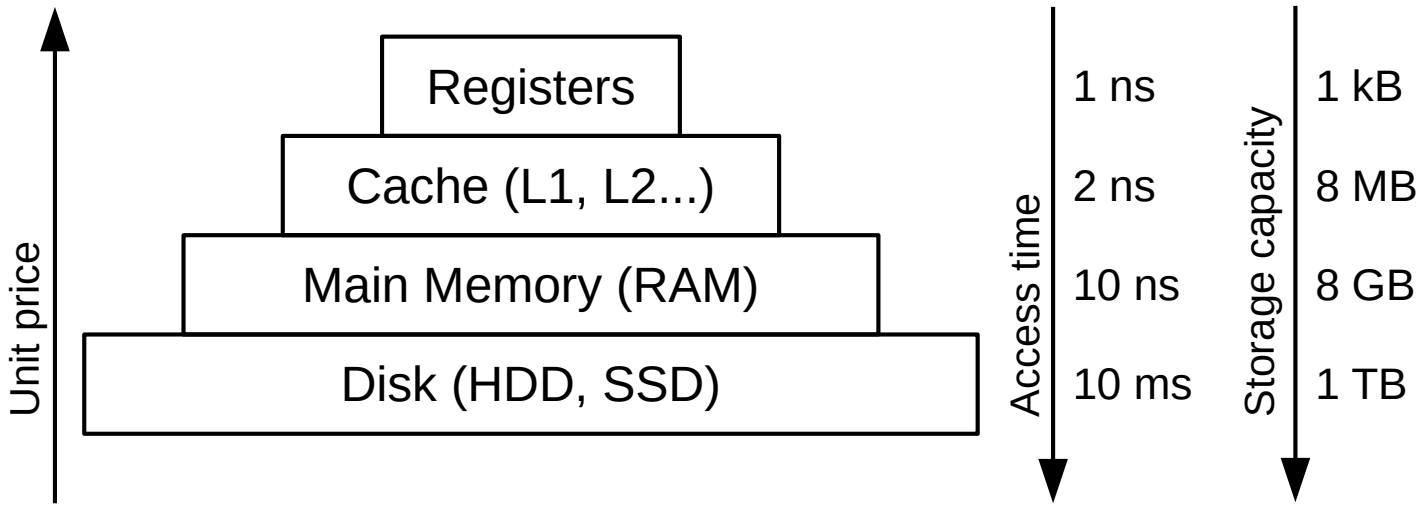
- odloženie kontextu bežiaceho procesu (t.j. registre a aktuálny stav procesora) v prípade prerušenia.
- komplikovaný pipeline procesora komplikuje návrh OS. (zmena usporiadania inštrukcií, spracovanie viacero inštrukcií v čase, ...)
- ak procesor podporuje multithreading OS musí tiež implementovať túto podporu.
- ak procesor je viac jadrový návrh OS toto tiež musí zohľadniť.

Pamäť


Monitor
Keyboard
USB printer
Hard disk drive
CPU
MMU
Memory
Video controller
Keyboard controller
USB controller
Hard disk controller
Bus

Pamäť

- Slúži ako úložisko pre všetko potrebné pre vykonávanie úloh výpočtového systému.
- Úložisko pre vykonávaný program, vstupné a výstupné dáta.
- Z hľadiska ceny, času prístupu a množstva ju môžeme členiť na:



Pamäť - Registre

- Priama súčasť jadra procesora.
- Okamžitý prístup k dátam.
- Vysoká cena - obmedzené množstvo registrov.
- Obsadenie registrov riadi kompilátor.

Pamäť - Cache

- Pridružená veľmi rýchla pamäť.
- Na procesoroch je bežné mať niekoľko úrovní takejto pamäte.
- V Cache sa ukladajú tie najčastejšie používané časti dát a programov.
- Vo väčšine prípadov je Cache riadená výlučne HW.
- Rozdelená na tzv. Cache Lines alebo Blocks.
- Bežnými problémami, s ktorými sa stretávame v Cache sú:
  - Kedy sa vkladajú dáta do Cache.
  - Do ktorého Cache Line sa dáta odložia.
  - Výber obete ak je Cache plná.
  - Kam v pamäti sa má obeť odložiť.
  - Ako sa propagujú zmeny/zápisy do dát v Cache.

Pamäť - Cache

L1 Cache

Dosahuje rýchlosť prístupu prakticky totožnú ako ku registrom. Jej veľkosť v procesore dosahuje zhruba $256kB$ pričom je rozdelená na pamäť programu a pamäť dát.

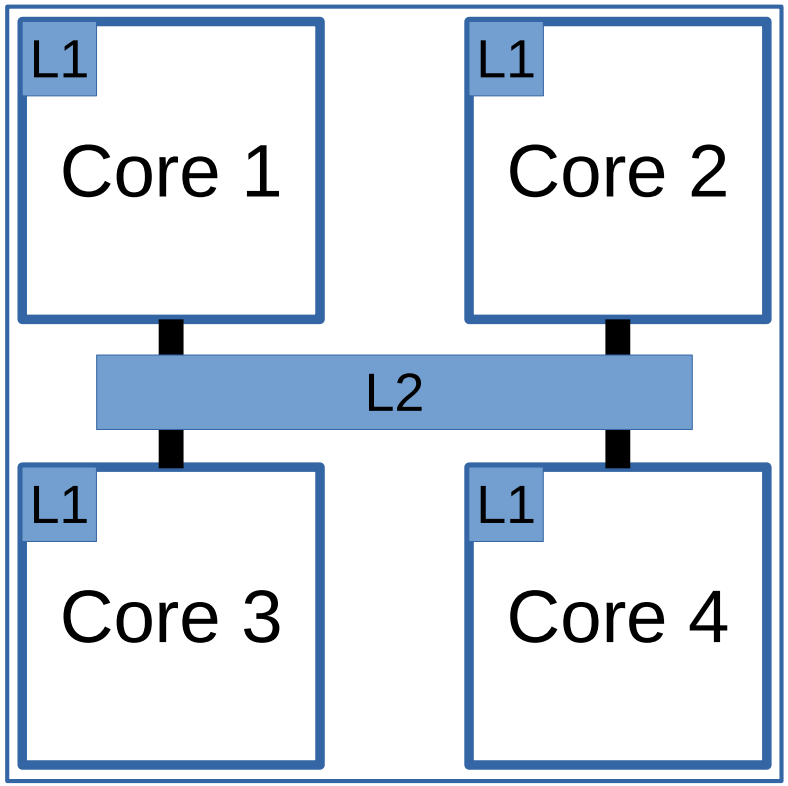
L2 Cache

Dosahuje rýchlosť prístupu 1-2 cykly procesora. Jej veľkosť v procesore dosahuje zhruba $8MB$ zvyčajne sa používa len na odloženie dát.

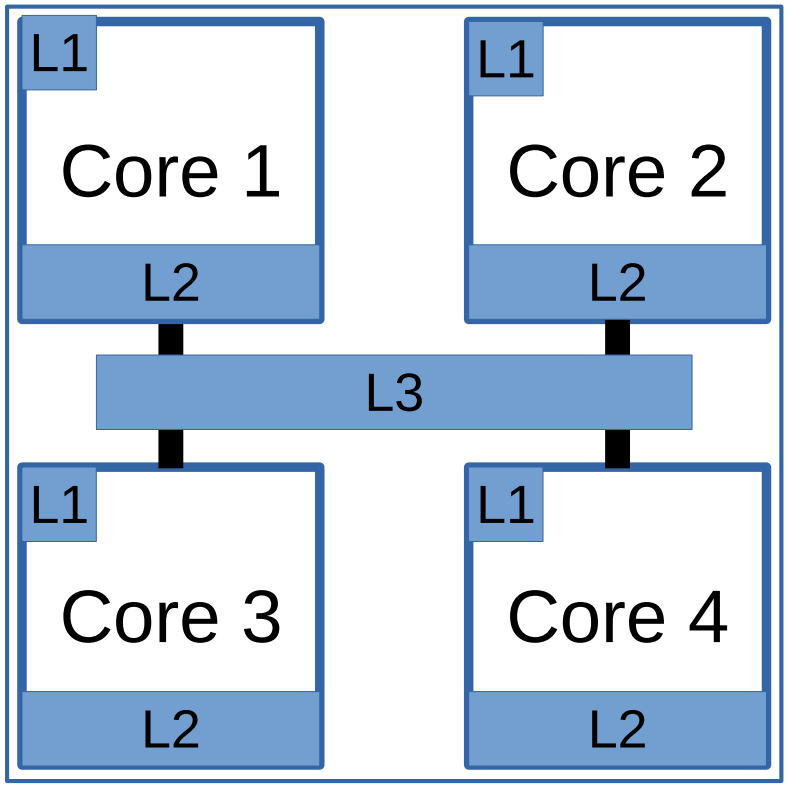
Cache Hit a Cache Miss

CPU udržuje asociatívnu pamäť odložených Cache Lines. CPU pri prístupe k dátam konzultuje túto pamäť. Ak existuje záznam v Cache máme tzv. Cache Hit. V prípade, že dáta nie sú v Cache, procesor musí pristúpiť do hlavnej pamäte. V takom prípade hovoríme o Cache Miss.

Pamäť - Cache



(a)



(b)

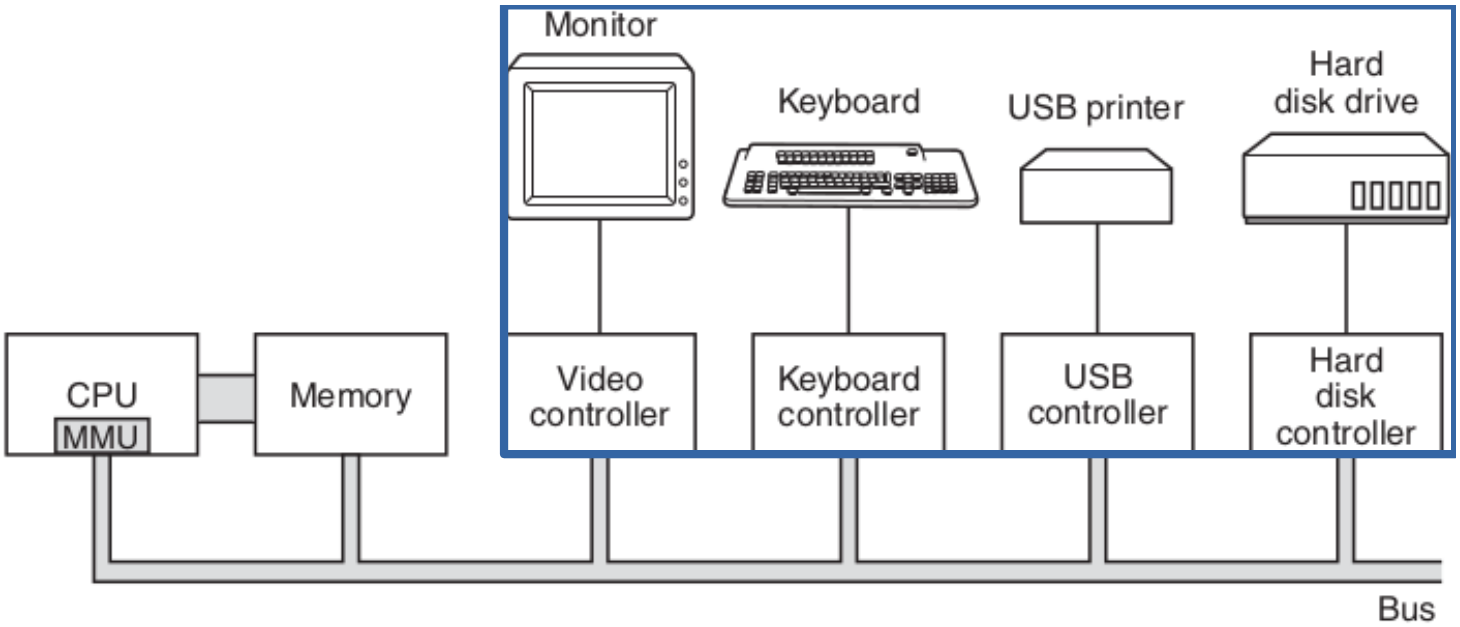
Pamäť - Main memory

- Centrálna pamäť počítača typu RAM.
- V súčasnosti dosahuje veľkosti presahujúce 8 GB.
- Pripojená k CPU prostredníctvom samostatnej zbernice.
- Prístupová doba rádovo väčšia ako pri Cache.
- Keďže aj RAM je bežne nedostatok riešime rovnaké problémy ako pri Cache.
  - Kedy sa vkladajú dáta do hlavnej pamäte.
  - Výber obete ak je hlavná pamäť plná.
  - Kam sa má obeť odložiť (disk?).
  - Ako sa propagujú zmeny/zápisy do dát v hlavnej pamäti.

Pamäť - Disk

- To čo sa nezmestí do RAM sa ukladá na diskoch.
- Dáta na diskoch sú organizované v blokoch.
- Prístupová doba k dátam je niekoľko rádov dlhšia ako do RAM.

I/O zariadenia



I/O zariadenia

- Zariadenia umožňujú skutočnú prácu s PC. (klávesnica, myš, monitor, WIFI)
- Zariadenie je ovládané Device Controller-om. Je to HW, ktorý je pripojený cez zbernicu k procesoru.
- Device Controller obsahuje registre, ktoré slúžia na jeho ovládanie.

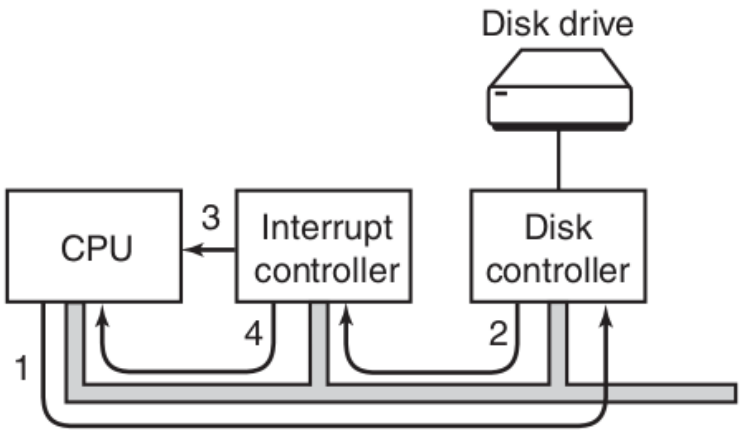
I/O zariadenia

Device Driver (Ovládač zariadenia)

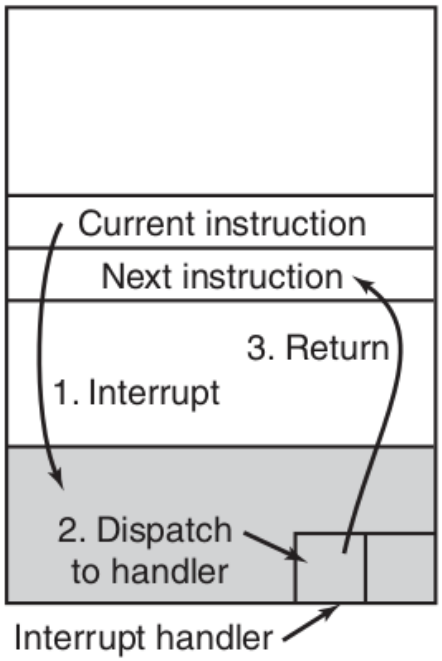
OS používa na prácu so zariadeniami tzv. Ovládače. Ovládač je abstrakciou, ktorá zjednodušuje prácu so zariadením. Ovládač možno implementovať:

- Pooling (Busy Waiting) - Používateľ nastaví zariadenie a v cykle kontroluje či zariadenie skončilo spracovanie.
- Interrupt - Používateľ nastaví zariadenie a pokračuje v inej činnosti. Zariadenie keď ukončí spracovanie generuje prerušenie.
- DMA (Direct Memory Access) - Používateľ nastaví zariadenie a určí kde sa odkladajú dáta a pokračuje v inej činnosti. Zariadenie ukladá dáta keď skončí generuje prerušenie.

I/O zariadenia - prerušenie



(a)



(b)

Stavebné kamene OS

Stavebné kamene OS

- Procesy (a vlákna)
- Adresný priestor
- Súborové systémy
- Používateľské rozhrania
- Systémové volania
- Manažment I/O

Procesy

Program

je súbor obsahujúci množinu usporiadaných krokov, inštrukcií vedúcich k dosiahnutiu želaného cieľa, správania.

Proces

je vykonávaný program s pridelenými prostriedkami výpočtového systému.

- Proces má pridelený adresný priestor, registre, súbory...
- OS udržuje informácie o procesoch v tzv. Process Table.
- Proces môže vzniknúť iba prostredníctvom iného procesu. Vzniká tzv. Process Tree.
- Procesy navzájom inter-reagujú. Interakcia je riešená OS.

Adresný priestor

Adresný priestor

je všetka potenciálna pamäť adresovateľná adresno-dátovou zbernicou procesora.

Fyzický Adresný priestor

je všetka reálne existujúca pamäť adresovateľná adresno-dátovou zbernicou procesora.

- Adresný priestor je väčšinou ďaleko väčší ako máme reálne k dispozícii.
- Jednotlivé procesy musia byť navzájom izolované.
  - Existuje ochranný HW, ktorý bráni zásahu do pamäte procesu.
  - Tento HW riadi OS.
- Proces teoreticky môže využiť celý adresný priestor.

Adresný priestor

Virtuálna pamäť

je koncept OS zabezpečujúci prístup k celému adresnému priestoru v limitovanom fyzickom adresnom priestore.

- Bežný súčasný procesor môže mať adresný priestor s veľkosťou do 2^{64} B (2^{30} * 16 GB).
- Bežné PC má RAM o veľkosti do 2^{34} B (16 GB)

Súborové systémy

File (Súbor)

je abstrakcia OS, ktorá umožňuje prácu s dátami používateľa bez nutnosti bližšie poznať zariadenie na, ktorom sú dáta uložené. Súbor je vlastne štruktúra, ktorá obsahuje informácie o blokoch disku, ktoré mu patria. Súbor môže byť vytvorený, zmazaný, otvorený, zavretý, čítaný a zapisovaný. Všetky operácie nad súborom implementuje OS prostredníctvom systémového volania.

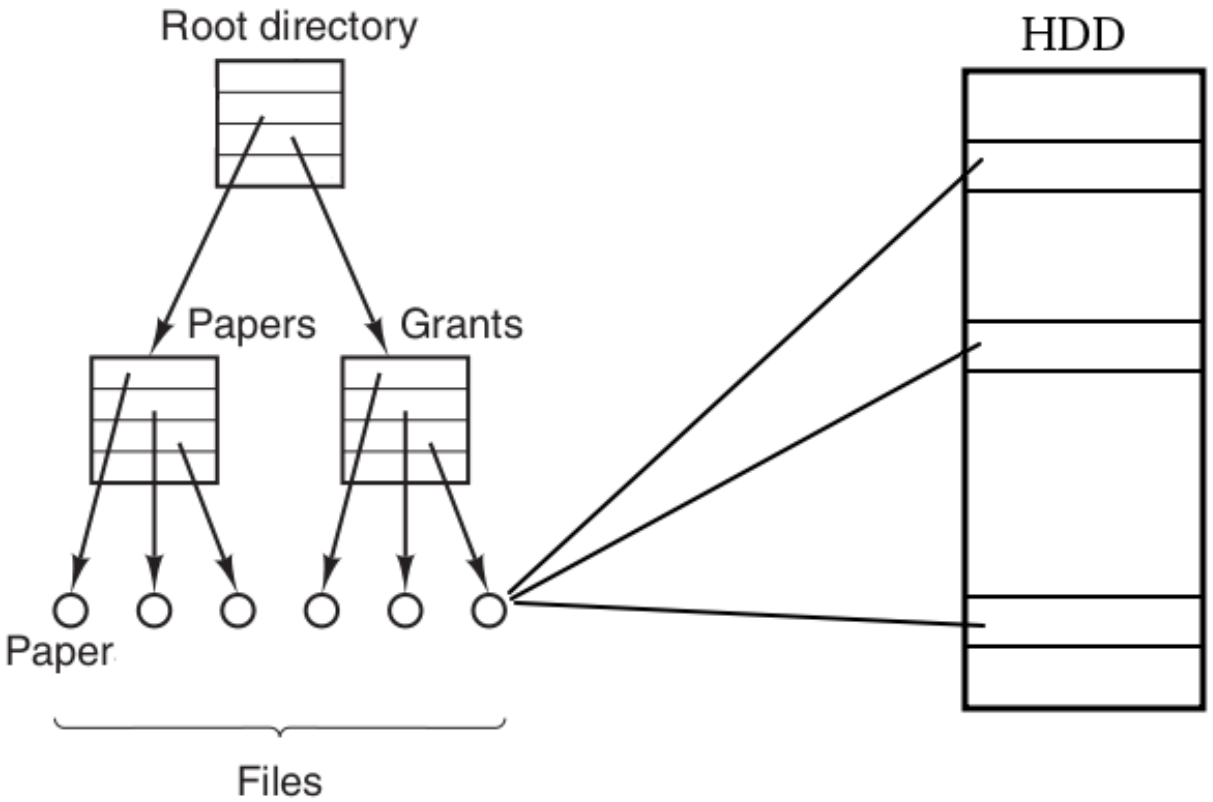
Directory (Adresár)

je vo všeobecnosti špeciálny súbor, ktorý obsahuje informáciu o všetkých súboroch a iných adresároch, ktoré sú pod nim vedené.

File System (Súborový systém)

je štruktúra adresárov s uloženými súbormi. Z pohľadu OS je File System abstrakciou so všeobecne známym API.

Súborové systémy



Shell a GUI

Shell

je program určený sprístupňujúci funkcie OS. V zásade sa jedná o príkazový riadok, kde používateľ zadáva príkazy v textovej podobe a dostáva odpovede rovnako ako text. Shell nemožno brať ako súčasť OS ale bez neho by sme si nepomohli. Shell definuje jednoduchý spôsob ako spúšťať ďalšie procesy a presmerovávať dáta medzi procesmi.

GUI

Grafickou nadstavbou OS. Na rozdiel od Shell je jeho reprezentácia komplikovanejšia ale na druhú stranu jeho ovládanie je pre bežného používateľa preferovanou voľbou. Rovnako ako u Shell nemožno ho považovať za priamu súčasť OS.

Systémové volania

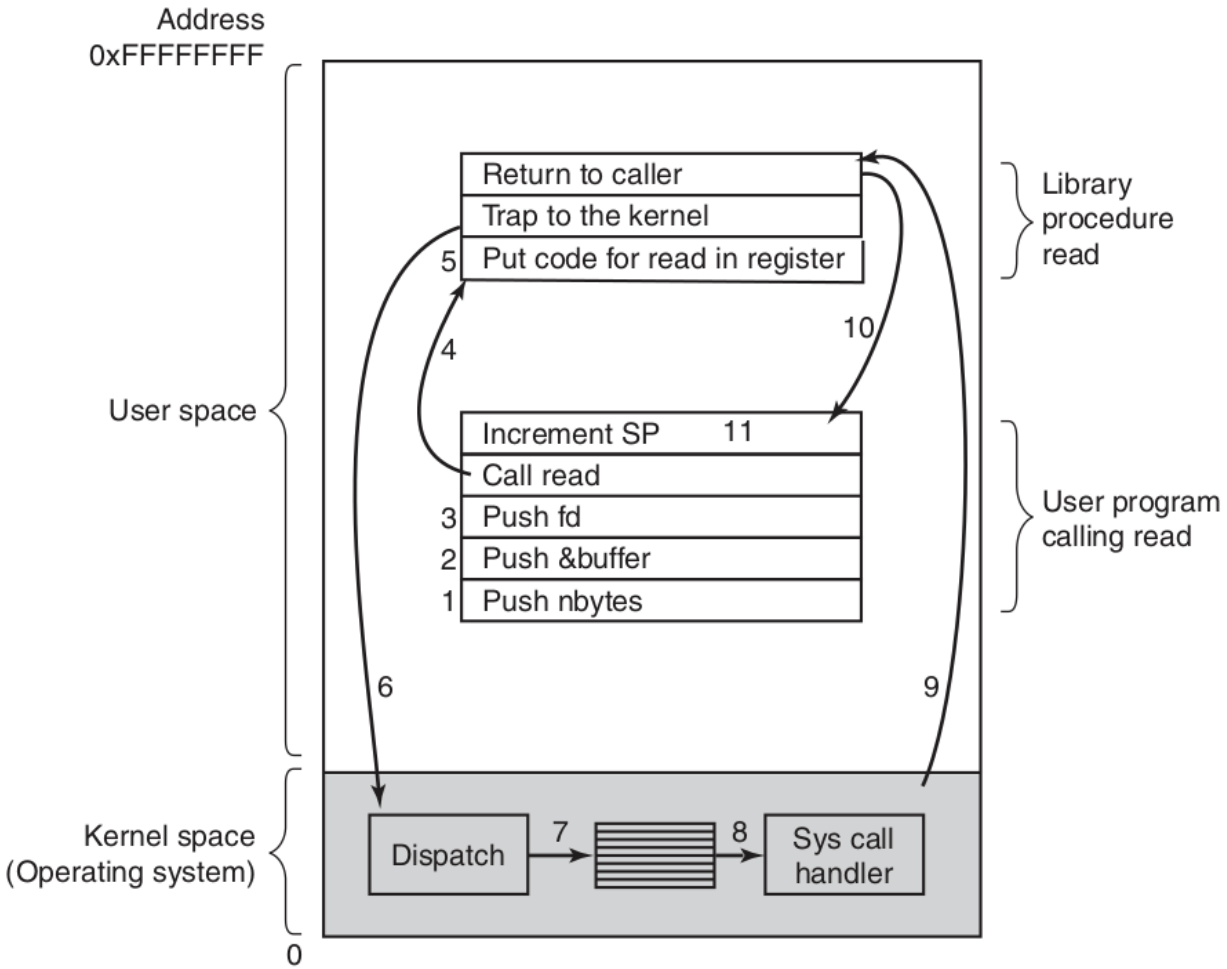
System Call (systémové volanie)

je špeciálna funkcia, ktorá je súčasťou abstraktného rozhrania (API) OS. Toto API bežne nájdeme v tzv. knižniciach OS. OS a HW vieme ovládať len prostredníctvom tohto API. Centrom systémového volania je vykonanie špeciálnej inštrukcie TRAP, ktorá zabezpečí prepnutie do privilegovaného režimu procesora. Po prepnutí do tohto režimu OS vyhodnotí aké systémové volanie sa zavolalo a následne ho vykoná.

POSIX

je štandard, ktorý presne vymedzuje minimálne rozhranie, ktoré by mal každý OS podporovať. Štandard určuje sadu systémových volaní a ich formu.

Systémové volania



Systémové volania

Process management

Call	Description
pid = fork( )	Create a child process identical to the parent
pid = waitpid(pid, &statloc, options)	Wait for a child to terminate
s = execve(name, argv, environp)	Replace a process' core image
exit(status)	Terminate process execution and return status

File management

Call	Description
fd = open(file, how, ...)	Open a file for reading, writing, or both
s = close(fd)	Close an open file
n = read(fd, buffer, nbytes)	Read data from a file into a buffer
n = write(fd, buffer, nbytes)	Write data from a buffer into a file
position = lseek(fd, offset, whence)	Move the file pointer
s = stat(name, &buf)	Get a file's status information

Directory- and file-system management

Call	Description
s = mkdir(name, mode)	Create a new directory
s = rmdir(name)	Remove an empty directory
s = link(name1, name2)	Create a new entry, name2, pointing to name1
s = unlink(name)	Remove a directory entry
s = mount(special, name, flag)	Mount a file system
s = umount(special)	Unmount a file system

Miscellaneous

Call	Description
s = chdir(dirname)	Change the working directory
s = chmod(name, mode)	Change a file's protection bits
s = kill(pid, signal)	Send a signal to a process
seconds = time(&seconds)	Get the elapsed time since Jan. 1, 1970

Zhrnutie

Zhrnutie

- Podoba OS úzko súvisí s HW ktorý zapúzdruje.
  - Návrh OS musí zohľadniť rôzne aspekty architektúry procesora.
  - Zariadenia a ich ovládanie sú veľmi pestré. OS musí poskytnúť vhodnú formu abstrakcie.
- Základnými konceptami s ktorými OS pracuje sú:
  - proces, strom procesov, tabuľka procesov
  - adresný priestor, virtuálna pamäť
  - súborové systémy, súbor, adresár, working directory
  - systémové volania, knižnice API OS
  - shell, GUI

Čo robiť do ďalšej prednášky

- Zoznamujte sa s prikazovym riadkom!!!!
  "ked chcete poznať príkazový riadok musíte myslieť ako príkazový riadok."
- Prečítať kapitoly 2.1, 2.2 a 2.4 z Tanenbauma.
  Pozrieme sa na procesy.