Úvod do informačních technologií

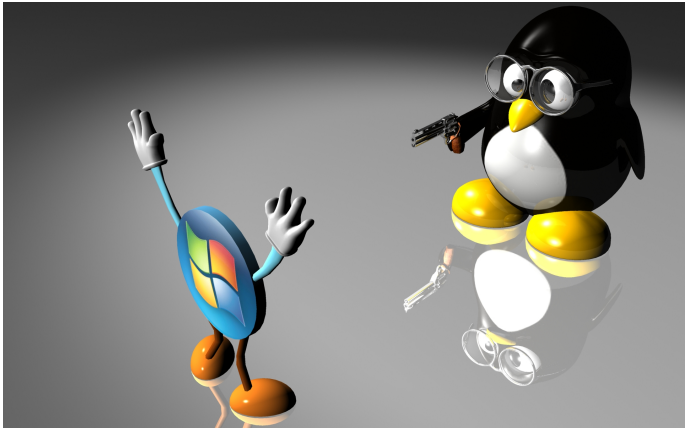
Jan Outrata

KATEDRA INFORMATIKY
UNIVERZITA PALACKÉHO V OLOMOUCI

přednášky

Operační systémy

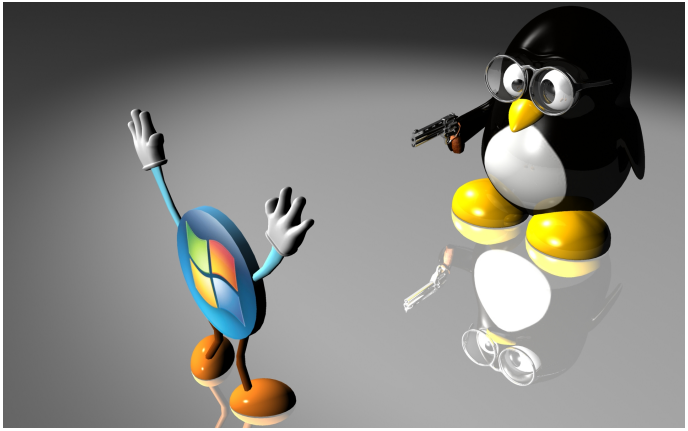
Co je to operační systém?



Operační systém je základní softwarové vybavení počítače, které se stará o správu systémových (hardwarových) zdrojů.

(autor neznámý)

Co je to operační systém?



Operační systém je základní softwarové vybavení počítače, které se stará o správu systémových (hardwarových) zdrojů.

(autor neznámý)

Úvod

Operační systém (Operating System, OS)

= základní softwarové vybavení počítače – rozhraní mezi uživatelem a počítačem

- umožňuje programům (aplikacím) běh na/v počítači pomocí programového rozhraní (API) a uživatelům práci s počítačem pomocí svého uživatelského rozhraní (UI) a programů
- cíl: snadné a efektivní využití počítače (pro uživatele i aplikace)
  - víceméně protichůdné požadavky – dříve důraz na efektivitu (a vůbec možnost), nyní spíše snadnost
  - kompromis, závisí na způsobu využití a typu počítače → různé OS
- poskytuje **abstrakci** (funkcí) hardware počítače, odstiňuje uživatele a aplikace od hardware, např. čtení souboru:
  - aplikace: otevření souboru zadaného (úplným) jménem a získání „objektu“ souboru, čtení souboru pomocí „objektu“ po bytech, zavření souboru
  - OS: nalezení režijních informací o souboru na základě jména a vrácení „objektu“ souboru, nalezení (čísel) sektorů disku s daty souboru a čtení sektorů, zrušení „objektu“ souboru

Úvod

Operační systém (Operating System, OS)

- abstrakce hardwarových zdrojů počítače: procesoru, operační paměti a vstupně/výstupních zařízení (viz von Neumannova koncepce počítače)
  - **dvě rozhraní**: pro komunikaci s hardware a pro umožnění využití hardware aplikacemi (API) a uživatelem (UI) skrze OS
  - **API (Application Programming Interface)** – zpřístupňuje služby OS programům, řešeno tzv. systémovými službami, dnes i virtualizované
  - **UI (User Interface)** – zpřístupňuje služby OS a programů uživatelům, textové příkazové (command line interface, CLI), grafické (GUI), virtualizované, mnohdy vícenásobné
- zajišťuje (bezpečnou a efektivní) **správu systémových (hardwarových) zdrojů** počítače
  - sdíleny běžícími programy v (krátkých) časových úsecích: procesor
  - rozdělovány mezi programy: paměť
  - dočasně programům přidělovány podle jejich potřeby: klávesnice, myš, I/O zařízení
  - virtualizovány pro transparentní sdílení programy: disková zařízení, grafický výstup, zvukový a síťový vstup/výstup aj., I/O zařízení

Operační systém (Operating System, OS)

- části:
  - **jádro (kernel)** – vlastní OS, monolitické (jeden program s veškerou funkcionalitou, příp. i s ovladači hardware) nebo tzv. mikrojádro (jen správa procesoru, paměti a komunikace mezi ostatními částmi realizujícími ostatní funkcionalitu a ovladače hardware), dnes i hybridní
  - **základní obslužné programy** – pro práci s OS a zdroji počítače, např. nástroje pro ovládání OS a práci s programy (aplikacemi) – **shell**, administrátorské a diagnostické nástroje pro práci s hardware a poč. sítí, základní nástroje pro manipulaci s daty aj.
  - **uživatelské rozhraní (UI)** – součást jádra OS, shellu nebo programy – záleží na použití a typu OS, neinteraktivní (úlohové/dávkové), interaktivní – textové s interpretem příkazů (shell) nebo grafické s (typicky) okenním systémem

Úvod

Operační systém (Operating System, OS)

- typy:
  - různé v závislosti na způsobu využití a typu počítače
  - → **univerzální** – pro desktopové a přenosné počítače typu **PC**, servery, mainframe apod.
  - **embedded** – specializované pro embedded zařízení, i upravené univerzální pro přenosné účelové počítače (např. Linux/Android, Apple iOS, MS Windows)
  - **reálného času** – zaručení vyřízení požadavku/odpovědi v pevně daném čase, např. VxWorks, QNX, upravené univerzální (např. RTLinux, MS Windows RT) i HW řešení, např. pro řízení strojů
  - **distribuované** – pro běh současně na více počítačích, simulace např. jedné společné paměti, pro počítačové **klastry (cluster)** = počítače propojené do sítě s možností běhu (typicky výpočetních) programů současně na všech
  - další ...
- dnes nejvíce používané: na desktopových PC MS Windows, Mac OS X, GNU/Linux, na síťových serverech unixové (GNU/Linux, BSD), MS Windows, na účelových počítačích různé (Linux/Android, Apple iOS, MS Windows), na embedded zařízeních různé (Linux, MS Windows)

Historie OS

- před 50. léty neinteraktivní ovládání počítače, bez OS, max. 1 program
- **multiprogramování** – od 50. let, více programů (dávek), dávkové OS $\Rightarrow$ potřeba přidělování paměti programům = správa operační paměti
- **jednoúlohové (single task)** – max. 1 spuštěný program (hlavní úloha), po dokončení nebo pozastavení další (vedlejší/doplňkové)
- **víceúlohové (multi task)** – od 60. let, více běžících úloh střídajících se při čekání na hardware (disk, periferie, obecně I/O) $\Rightarrow$ potřeba plánování úloh = správa procesoru, samostatná činnost procesoru a ostatních zařízení $\rightarrow$ koncept přerušení
- **sdílení času (time-sharing)** – od 70. let, úlohám přidělován procesor na krátká časová kvanta $\rightarrow$ iluze současného běhu úloh $\Rightarrow$ potřeba (hardwarový) časovač ... dnešní OS
- **víceuživatelské (multi user)** – více uživatelů současně $\Rightarrow$ potřeba virtualizace uživ. rozhraní

Historie OS

- od 50. let pro mainframe počítače, např. **OS/360** (pro IBM System/360, správa hardware, 50. a 60. léta), SCOPE (CDC), MCP (Burroughs, virtuální paměť, 60. léta), GECOS (GE, úrovně oprávnění programů a uživatelů), **Multics** (MIT/GE/Bell Labs, víceuživatelský), TOPS (DEC, 70. léta), **Unix**, VMS (DEC VAX), SunOS/Solaris (Sun), z/OS (IBM), **Linux**
- od 70. let pro mikropočítače, první minimalistické v ROM (tzv. monitory), diskové jednoúlohové např. CP/M (Digital Research), **MS DOS** (Microsoft, IBM PC, PC DOS, Free DOS), od 80. let víceúlohové pro PC např. **MS Windows** (Microsoft), NeXTSTEP (NeXT), **Mac OS X** (Apple, 90. léta), **Linux**, se zvyšováním výkonu i OS pro mainframe (Unix, Solaris)

Příklady OS (pro PC)

Unix

- 1965 Multics, konec 60. let, 1971, Bellovy laboratoře, verze System V, BSD (Berkeley System Distribution), AIX (IBM), HP-UX (HP), SunOS/Solaris (Sun) aj., volně použitelné i proprietární
- víceúlohový, víceuživatelský
- architektura: jádro + shell + programy (i implementující textové i grafické UI)
- pro různé procesory a počítače
- inspirující a ovlivňující vývoj dalších OS

Další: Minix (výukový), OS/2 (IBM, ukončený), Hurd (GNU), Plan 9 (Bell Labs, experimentální), Android (Linux, Google), iOS (Apple), Chrome OS (Linux + webový prohlížeč Google Chrome), Firefox OS (Linux, Mozilla), RTOS1 (real-time) aj.

Příklady OS (pro PC)

DOS

- 1981, Microsoft, IBM, Digital Research, různé proprietární verze
- jednoúlohový, jednouživatelský
- architektura: jádro + shell + programy (i implementující textové UI)
- pro procesory Intel 80x86

MS Windows

- 1993 (NT), Microsoft (od 1985 (1983?) pouze jako víceúlohová GUI nadstavba nad DOS), proprietární
- víceúlohový, jednouživatelský (řada 9x, dříve), víceuživatelský (řada NT, dnešní)
- architektura (NT): jádro + subsystémy emulující API jiných OS (DOS, starší MS Windows, částečně unixové, OS/2) + shell (implementující GUI) + programy
- pro procesory Intel 80x86, ARM, Aplha (dříve)

Příklady OS (pro PC)

Mac OS X

- 1999 (Server, 2001 desktop), Apple (od 1984 Mac OS), pro počítače Apple Macintosh (Mac), proprietární
- víceúlohový, víceuživatelský, unixový
- architektura: jádro XNU (BSD Unix/Mach) + shell (API) + GUI + programy
- pro procesory Intel 80x86, ARM, IBM PowerPC (dříve)

GNU/Linux

- 1991 (1983 GNU), Linus Torvalds, svobodný (free) software (licence GNU GPL) ve formě distribucí („balení“ s dalšími obslužnými programy ve formě balíčků, i tzv. “živé”)
- víceúlohový, víceuživatelský, unixový
- architektura: jádro Linux + shell (GNU aj.) + programy (i implementující textové i grafické UI)
- pro mnoho procesorů, počítačů (i mimo PC) a jiných zařízení (elektronika)

Správa procesoru

Vykonávání instrukcí

- program = sekvence (binárních) kódů instrukcí, registrů procesoru a dat (čísla, texty, hodnoty adres do operační paměti a vstupně/výstupních zařízení)
- stejná (RISC) nebo proměnná (CISC) délka kódů instrukcí – 1 až 4? byty
- **operandy** = parametry instrukcí, registry a data, specifický počet (obvykle 0 až 2), přípustné kombinace pro každou instrukci
- výsledek instrukce často ukládán do prvního operandu
- vykonávání instrukce
  - trvá určitý počet **taktů/tiků** (na vnitřní frekvenci procesoru), jednotky až stovky
  - až 7 fází: např. načtení, dekódování, načtení operandů, provedení, uložení výsledku
  - **pipelining** – částečně paralelní provádění instrukcí (různých fází), nelze vždy, např. kvůli instrukcím podmíněných skoků (= implementace podmínek a cyklů v programu)

Správa procesoru

Vykonávání instrukcí

- vykonávání instrukce
  - **superskalární architektura** – více duplikovaných částí procesoru, např. ALU, částečně paralelní provádění instrukcí (i stejných fází), použití i na podmíněné skoky (vykonávání obou větví, po skoku i bez skoku, současně, u neplatné se pak ukončí – tzv. speculative execution, nebo předvídání podmínky skoku/správné větve – tzv. branch prediction) $\rightarrow$ instrukce může ve výsledku v průměru trvat pod 1 takt
  - **vícejádrové procesory a více procesorů** – (plně) paralelní provádění instrukcí, symetrické a nesymetrické architektury (hardware i OS)
- sekvenční pořadí vykonávání instrukcí tak, jak jsou v programu (viz von Neumannova architektura)
  - registr **EIP** – adresa následující instrukce, automatické zvětšování
  - výjimky = jiné změny EIP: instrukce skoků (na adresu), volání podprogramů (funkcí, procedur, metod objektů apod.) a obsluh přerušení + návrat na místo volání/přerušení

Správa procesoru

Jazyk symbolických adres (“assembler”)

= jazyk (textově) pojmenovaných instrukcí, např. MOV, ADD, MUL, AND, CMP, JE, JMP, a registrů procesoru, (zápisů) čísel a textu, hodnot adres, proměnných atd.

```
MOV eax, promenna1; CMP ebx, promenna2; JE adresa
```

- překládán do kódů instrukcí
- **přímá a nepřímá adresa do paměti** – adresa vypočítána z hodnot v registrech a zadaných přímo, např. posunutí + báze + index×faktor, použití např. pro přístup do strukturovaných dat, k lokálním proměnným apod.

Vyšší programovací jazyky

- vyšší úroveň abstrakce, např. iterace přes prvky seznamu → cyklus průchodu strukturovanými daty → jména instrukcí → kódy instrukcí
- **překladač** – přeloží (přepíše) program z jednoho (vyššího) prog. jazyka do jiného (nižšího) jazyka, typicky až do kódů instrukcí
- **interpret** – přeloží program z prog. jazyka do příkazů interní formy a tyto vykoná

Správa procesoru

Přerušení (Interrupt)

- původně pro řešení komunikace (rychlého) procesoru s (pomalými) I/O zařízeními, běžícími samostatně:
  - dříve procesor: vyslání požadavku na zařízení, **aktivní čekání** na vyřízení (= smyčka testující stav oznamující vyřízení), pokračování ve výpočtu

  → vyslání požadavku, pokračování ve výpočtu zatímco zařízení zpracovává požadavek, oznámení vyřízení požadavku = **přerušení procesoru**
  - např. procesor vyšle požadavek čtení sektoru z disku (dá požadavek s číslem sektoru na sběrnici) a pokračuje ve výpočtu, disk najde sektor, načte do své cache a vyvolá přerušení, procesor vyšle požadavek zaslání dat, disk pošle, procesor uloží do operační paměti, požadavek na další data atd.

= pozastavení vykonávání programu, vykonání programu tzv. **(rutiny) obsluhy přerušení** implementované OS (např. ovladači zařízení), pokračování vykonávání programu

- během vykonávání obsluhy přerušení další přerušení zakázána nebo systém **priorit přerušení**

Správa procesoru

Přerušení (Interrupt)

- **hardwarová**: přídavné karty (dříve), disková zařízení (dříve), vstupně/výstupní zařízení, HW časovač aj., 256 přerušení u Intel 80x86
- **softwarová** – vyvolána OS pro vlastní potřeby fungování nebo programy při volání služeb OS (tvz. **systémová volání**)

DMA (Direct Memory Access)

= způsob přenosu dat mezi zařízením a operační pamětí přímo, bez řízení procesorem, pro větší množství dat, např. pro disková zařízení

- procesor pouze naprogramuje **řadič DMA** a vyšle prvotní požadavek, zbytek řeší řadič

Mapování paměti

... zařízení do operační paměti, např. přídavné karty

- přímý přístup do paměti zařízení skrze přístup do operační paměti

Správa procesoru

Proces

= spuštěný program

- vzniká spuštěním programu – přidělení systémových zdrojů (paměť, procesor aj.), načtení kódu do paměti, spuštění
- **životní cyklus** (stavy): vytvořený, připravený (k plánování, ready), čekající (standby), běžící, blokovaný, odložený, ukončený aj.

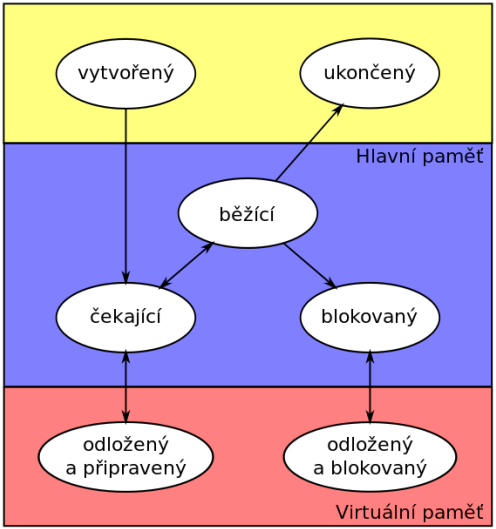
Obrázek: Životní cyklus procesu

- vztah rodič-potomek – na některých OS jedna hierarchie, např. unixových

Plánování běhu procesů

- přidělování procesoru procesům (přepínání procesů) na vymezené **časové kvantum** (time-sharing) – různě dlouhé u různých OS a určení OS, nastavitelné, např. 10 ms (MS Windows), 1–100 ms (Linux)
- řešené situace: přidělování procesoru procesům a procesů na procesory

Správa procesoru



Správa procesoru

Plánování běhu procesů

- provádí **plánovač OS (scheduler, dispatcher)** – vyvoláván při ukončení/pozastavení běhu procesu: ukončením procesu, čekání (na hardware), volání služby OS, vypršení přiděleného kvanta času
- vypršení kvanta – **přerušení od časovače**, možnost prodloužení (interaktivita)
- **strategie**: cyklická obsluha (round-robin), systém priorit, férové přidělování kvanta aj.
- **systém priorit**: idle, normální, vysoká, realtime, aj., dědění, zvyšování (interaktivita, konec čekání na hardware, synchronizace) atd., např. -20 až 20 (unixové), 0 až 31 (MS Windows)
- **kooperativní** – pouze dobrovolné ukončení běhu procesu ukončením, čekáním, voláním služby OS, dříve (spíše u jednoúlohových)
- **preemptivní** – i "násilné" ukončení běhu procesu ukončením kvanta, přepnutí na proces s vyšší prioritou (preempce), dnes i preempce částí jádra (kernel preemption), např. ovladačů zařízení

Správa procesoru

Plánování běhu procesů

- **režimy běhu procesů**: uživatelský (user) a jaderný (kernel) – jádro OS samotné (dříve u jednoúlohových OS i procesy), procesy během vykonávání systémových služeb ($\rightarrow$ přepnutí kontextu), podpora procesoru (reálný a chráněný režim, Ring 0–3, privilegované instrukce aj.)
- **symetrický multiprocesoring (SMP)** – plánování na více rovnocenných procesorů, **afinita procesu** = povolení/zakázání běhu procesu na vybraných procesorech

Správa procesoru

Vlákno (Thread)

$\sim$ vlákno vykonávání (thread of execution)

$=$ sekvence instrukcí plánovaná procesorem (podobně jako proces)

- typicky v rámci procesu, může jich být v procesu víc $\rightarrow$ program „dělá víc věcí zároveň“
- realizace: instrukce vlákna = podprogram, **sdílená paměť** a jiné zdroje procesu programu, implementace v programové knihovně a/nebo v OS
- plánování:
  - implementace – modely 1:1, 1:N, M:N
  - systém priorit (relativně k procesu), afinita – obvykle nepřesouvat vlákno na jiný procesor (kvůli cache)
  - v MS Windows NT a (některých plánovačích) Linuxu plánována pouze vlákna

Správa procesoru

Komunikace a synchronizace

- pro procesy a pro vlákna, ale i pro samotný OS!
- procesy jsou paměťově oddělené (každý má svoji přidělenou paměť) – komunikace pomocí speciální sdílené paměti a **posíláním zpráv**
- vlákna sdílejí paměť procesu – komunikace pomocí sdílené paměti

→ soupeření (race) o sdílenou paměť a jiné zdroje počítače, zvláště při více procesorech

- **chyba souběhu (race condition)** = chybné pořadí běhu (procesů a vláken) vedoucí k nekonzistetním stavům při konfliktních operacích, např. čtení-zápis – fáze operací (i instrukcí procesoru) např. čtení z paměti, operace, zápis do paměti

Správa procesoru

Komunikace a synchronizace

- potřeba **synchronizace** = určení specifického pořadí běhu (procesů a vláken)
  - **atomické operace** = nedělitelné, nepřerušitelné, sekvenčně prováděné (ty konfliktní), např. nastavení nebo inkrementace atomické proměnné, použití pro implementaci tzv. synchronizačních primitiv
  - **synchronizační primitiva**: zámek (mutex), semafor (počítadlo), kritická sekce, událost, monitor a další
  - hardwarová podpora: atomické instrukce procesoru (test-and-set, fetch-and-add, compare-and-swap aj.), zakázání přerušení (při jednom procesoru), preempce (při více procesorech)
  - softwarové implementace: Dekkerův (1965), Petersonův (1981) algoritmus, vyžadují (pouze) atomické uložení hodnoty do proměnné
  - implementovaná a poskytovaná OS, ale i využívaná v rámci samotného OS!

Komunikace a synchronizace

- **uváznutí (deadlock)**
  - = vzájemné čekání na výlučně vlastněné zdroje (např. chráněné zámky) při modelu využívání "požadavek na přivlastnění-používání-uvolnění"
  - podmínky vzniku: 1. výlučné vlastnictví, 2. čekaní při vlastnictví jiného, 3. vzájemné (cyklické) čekání, 4. nemožnost preempce (násilného odebrání prostředku)
  - řešení deadlocku: neřešení (ignorování), detekce a zotavení, prevence (zamezení vzniku, tj. nenaplnění podmínek), vyhýbání se (přidělování prostředků tak, aby nenastaly podmínky)
  - dnešní OS neřeší (ignorují) – proces(y) v deadlocku ukončí uživatel

Správa operační paměti

= přidělování paměti samotnému OS a procesům a uvolňování, evidence volné a přidělené paměti

- **kooperativní** – dříve u jednoúlohových OS, volitelná programem, program může používat i jinou (celou, cizí, i OS), např. při chybě
- **ochrana přidělené paměti** – procesy "nevidí" do paměti jiného procesu, podpora hardware, kontrolováno OS, např. segmentace, stránkování
- paměť fyzicky adresována lineárně – fyzická adresace neumožňuje relokaci dat (programu) $\rightarrow$ **logická adresace** paměti programu – na fyzickou adresu přepočítává procesor ve spolupráci s OS (dříve i programem)
- realizace **virtuální paměti**

Správa operační paměti

Přidělování souvislých bloků

- před logickou adresací paměti
- stejně velké → **vnitřní fragmentace** (nevyužité místo v bloku), např. IBM/360
- malé proměnlivé délky → **vnější fragmentace** (po uvolnění nedostatečné místo pro větší blok), např. MS DOS
- algoritmy výběru bloku: first fit, best fit apod.

Správa operační paměti

Segmentace

- **segment** = log. kus paměti programu (pro instrukce, data aj.) určený lineární adresou (**báze**), délkou (64 kB, 4 GB), oprávněními přístupu (DPL, Ring 0–3, 0 pro OS, 3 pro procesy, při porušení přerušení), aj.
- segmenty se mohou (i úplně) překrývat (pro paměť OS se překrývají)
- logická adresa má dvě složky: báze segmentu a posunutí (**offset**) v segmentu
- lineární adresa = složení báze a offsetu, např. součet
- báze zadaná přímo (reálný režim procesoru) nebo součástí tzv. **deskriptoru segmentu** vybraného v tabulce deskriptorů pomocí **selektoru** (chráněný režim procesoru)
- segmenty a tabulky deskriptorů segmentů spravuje OS, dnes triviálně – programům stačí offset ($\sim$ lineární adresa) = režim flat, např. s přidělenou pamětí 4 GB v/na 32-bitovém OS/procesoru
- výpočet lin. adresy provádí procesor: báze/selektor v k tomu určeném registru (segmentový, např. CS, DS, SS), lineární adresa tabulky desktriptorů ve speciálním registru (GDTR, LDTR)

Správa operační paměti

Stránkování (Paging)

- nezávislé na segmentaci
- lineární adresový prostor paměti programu disjunktně rozdělen na kousky = **stránky (page)**, délka typicky 4 nebo 8 kB (ale i třeba 1 nebo 4 MB, závisí na procesoru)

= stránky mapované 1:1 na stejně velké kousky operační paměti = **rámce (frame)** adresované fyzickou adresou – stránky za sebou nemusí být (a nejsou) mapovány na rámce za sebou

- mapování pomocí **hierarchie tabulek (tzv. adresářů)** udržované OS – části lin. adresy pro adresaci tabulek (10 bitů) na různých úrovních (až 4), rámce v tabulce (10 bitů) a offset (12 bitů) v rámci, velikost tabulky = velikost stránky
- výpočet mapování provádí procesor
- přístup do paměti = čtení tabulky a pak rámce paměti, zrychlené pomocí **TLB cache** v procesoru – princip lokality (data pohromadě = cache hit)

Správa operační paměti

Stránkování (Paging)

- práva přístupu, zápisu, vykonávání kódu programu (tzv. NX bit) ke stránce, značka neplatnosti aj. – při porušení přerušení
- umožňuje realizovat virtuální paměť
- **copy-on-write** – sdílení stránek (bloků) paměti se stejným obsahem mezi procesy, nová samostatná až při změně, využití: sdílená paměť mezi procesy např. pro kód sdílených knihoven programů, úspora paměti, realizace: výpadek stránky pouze pro čtení při zápisu, duplikace s povolením zápisu, zápis
- netrpí vnější fragmentací

Správa operační paměti

Virtuální paměť

= (logická) paměť pro procesy realizovaná pomocí více různých pamětí počítače (RAM, pevný disk atd.), typicky větší než (fyzická) operační paměť

- samotný OS (jádro) pro sebe (typicky) nepoužívá, má vlastní správu paměti
- **swapovací prostor** = část virtuální paměti jinde než v operační paměti (RAM), typicky na pevném disku
- **odstránkování paměti** = přesun stránek paměti z operační paměti do swapovacího prostoru („swapování“), důvod: programy nevyužívají pořád všechen kód a data, při nedostatku místa v operační paměti **algoritmy výběru oběti** FIFO (nejstarší), LFU (nejméně často použitá), (pseudo-)LRU (nejdéle nepoužitá) aj.
- **zamykání paměti** – stránky, které nelze odstránkovat, např. (části) OS, programové obsluhy (HW) přerušení, paměť programu mapovaná z jiných zařízení apod.

Správa operační paměti

Virtuální paměť

- stránkování na žádost (on demand paging):
  - aktuálně vykonávaný kód a používaná data programu musí být v operační paměti, když nejsou → **výpadek stránky (page fault)** = přerušení instrukce, načtení (s případným swapováním při nedostatku místa), opakování instrukce
  - algoritmy alokace minimálního počtu rámců procesu (desítky až stovky), trashing, pracovní množina rámců (v lokalitě programu) atd.
  - v obsluze výpadku stránky kontrola na platnost adresy, tj. existence stránky
- přidělování souvislých bloků stránek – např. **Buddy algoritmus** (spojování uvolněných)

Správa diskového prostoru

Oddíly/particie (volumes/partitions)

= disjunktní části prostoru, ve kterých se vytváří **souborový systém**

- různé formáty evidence, např. MS DOS/MBR tabulka particií (primární, rozšířené, logické), Sun tabulka oddílů, UEFI/GUID tabulka particií aj.
- vytvořené staticky (s pevnou velikostí) nebo dynamicky (s proměnnou velikostí)
- **Logical Volume Management (LVM)** – logické oddíly, dynamicky vytvářené, fyzicky uložené ve statických i na více discích, další vlastnosti RAID

Správa diskového prostoru

Souborový systém (File system)

- data na disková zařízení (do oddílů/particií) ukládána ve formě **souborů** = data + režijní informace souboru, např. umístění na disku (čísla sektorů), velikost, jméno, přístupová práva aj.

= způsob uložení souborů a jejich dat (rozdělení do bloků uložených do sektorů diskového zařízení)

- pro efektivní a spolehlivý přístup k datům
- **typy souborů**: soubor, adresář, dále např. odkazy, speciální (pro zařízení, roury aj.)
- **souborové operace**: vytvoření, otevření, čtení, zápis, změna aktuální pozice v otevřeném (za konec = zvětšení), zmenšení, uzavření, výmaz
- **přístupová práva**: čtení, zápis aj. (u různých OS, např. spouštění u unixových) pro různé uživatele a jejich skupiny
- **adresářová struktura** = log. (typicky) stromová struktura souborů

Správa diskového prostoru

Souborový systém (File system)

- dříve jeden, limitovaný (celková kapacita, omezení adresářové struktury, délka jména souboru apod.)
- dnes více pro různá zařízení, téměř bez omezení, na unixových OS abstrakce pomocí **virtuálního souborového systému (VFS)** – definuje jednotné API konkrétních
- zabezpečení proti chybám diskového zařízení: rezervace bloků pro vadné sektory zařízení, **žurnálování**, redundance (RAID), deduplikace, snímky (snapshot) atd.
- např. FAT (MS DOS, výměnné disky a přenosná záznamová zařízení), NTFS (MS Windows), HFS (Mac OS), UFS (UNIX, Solaris), ZFS (Solaris), Ext2/3/4, Reiser, JFS, XFS, Btrfs (Linux), ISO 9660 (CD), UDF (DVD), ...

Ovladače zařízení (Device drivers)

= programy (části OS) pro komunikaci s vstupně/výstupními zařízeními – přídavné karty (grafické, zvukové, síťové apod.), disková zařízení (pevné disky, mechaniky výměnných disků aj.), zařízení připojená k vnějším sběrnicím (USB, FireWire) atd.

- integrální (např. Linux) nebo volná součást OS (dodávaná samostatně výrobci zařízení, např. MS Windows)
- **abstrakce zařízení** pro OS a programy, transparentní použití
- implementují správu paměti zařízení, volání služeb zařízení a obsluhy HW přerušení od zařízení, dále např. plánování DMA přenosů apod.

Síťování (Networking)

= podpora komunikace v lokální a rozhlehlé počítačové síti (Internet)

= implementace síťových protokolů, na různých vrstvách (např. Ethernet, TCP/IP, DNS), tzv. **network stack**

- využití ovladačů síťových zařízení pro nízkoúrovňové služby
- **abstrakce síťového připojení** pro programy
- síťový OS = OS s podporou síťování

Bezpečnost

Vnitřní (lokální)

- důležitá u víceúlohových a víceuživatelských OS
- **operačního systému** – jaderný režim vykonávání služeb OS, privilegované instrukce procesoru, oddělené paměťové prostory programů a jádra OS atd.
- **programů** – výlučné přidělování procesoru, ochrana přidělené paměti, oddělení pamětí programů, synchronizace aj.
- **dat** – zabezpečení uložení (vč. např. redundance), práva přístupu aj.
- **uživatelů** – uživatelské účty + **autentizace**, přidělené zdroje (procesor, paměť, diskový prostor apod.) + **autorizace**, oddělení uživatelů, úrovně oprávnění (obyčejní uživatelé pro práci, správce pro administraci OS)
- u všech typů audit a logování
- úrovně zabezpečení OS

Bezpečnost

Vnější (síťová)

- důležitá u síťových OS
- zabezpečení proti neoprávněnému přístupu a využití zdrojů HW, OS a programů, primárně dat, ze sítě
- metody filtrace síťové komunikace a přenosů dat pomocí tzv. firewallu, skrývání (podsítí počítačů) a anonymizace (počítačů, uživatelů) v síti, systémy detekce a prevence proti útoku na bezpečnost ze sítě, autentizace programů, uživatelů, šifrování dat atd.

$\rightarrow$ počítačové sítě