Operační systémy (OS)

Pojem "Operační systém", jeho jádro a nadstavby, systémové a aplikační programy. Komponenty jádra OS. Pojem výpočetní proces, multiprocesní zpracování, plánování výpočetních procesů, jejich správa, vznik a zánik. Správa komponent výpočetního systému,. souborů a vstupních a výstupních zařízení.

1 Základní struktura OS

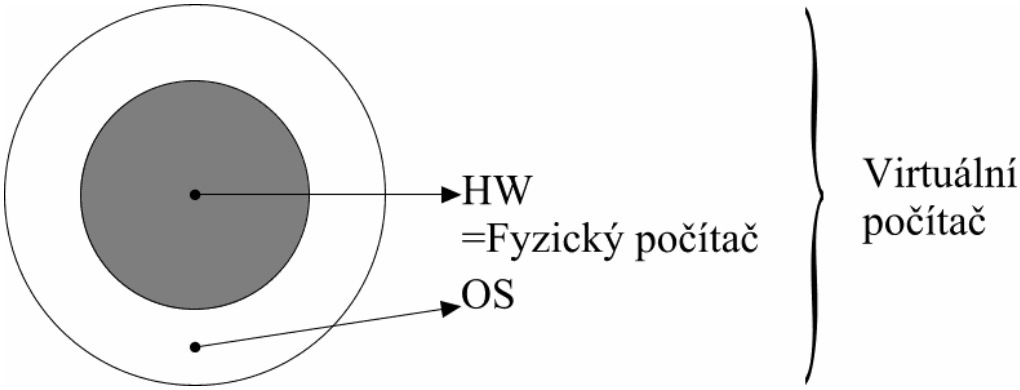
Operační systém je programové vybavení, jehož primárním účelem je správa prostředků daného výpočetního systému a jejich co nejefektivnější poskytování uživatelským programům. Operační systém musí:

- Mít přehled o všech dostupných prostředcích
- Definovat pravidla, na jejichž základě se rozhoduje o přidělení prostředku
- Přidělovat prostředky
- Vyžadovat navrácení přidělených prostředků

Prostředky výpočetního systému se dělí do několika základních skupin:

- procesory
- operační paměť
- souborový systém
- vstupní a výstupní zařízení

Správu těchto prostředků zajišťuje **jádro OS** – základní část, která je trvale přítomna v operační paměti. **Jádro OS** by mělo oddělit uživatelské programy od hardware (HW). Spolu s ním by mělo vytvořit tzv. **virtuální počítač** (obr. 1). Uživatelské programy volají služby OS na vyšší úrovni pomocí virtuálních instrukcí (volání OS – např. napiš znak, otevři soubor). Dosáhne se tím transparentní práce s rozdílným



obr. 1 Virtuální počítač

HW. Pokud k tomu existují HW prostředky, jádro je jediná část, která běží v tzv. **privilegovaném módu** procesoru a má přímý přístup ke všem zdrojům počítače. Ostatní programy musí používat volání jádra, jinak dojde kvyvolání přerušení a předání řízení OS.

Jako **systémové programy (procesy)** se označují funkce OS, které nejsou klíčové pro chod operačního systému. Tvoří další slupku nad virtuálním počítačem. Jedná se

např. o programy, zajišťující přihlašování uživatelů, interprety příkazů (shells) nebo různé síťové služby. Systémové procesy již nemusí mít charakter virtuálních instrukcí, tzn. po vyvolání neprovádějí přesně danou operaci danou parametry volání, jejich činnost může záviset na momentální situaci a pro svou práci využívají jádro OS. Obvykle se jedná buď o víceúčelové prostředky (editory) nebo systémově orientované programy (překladače, správa disků). Běží v uživatelském módu procesoru.

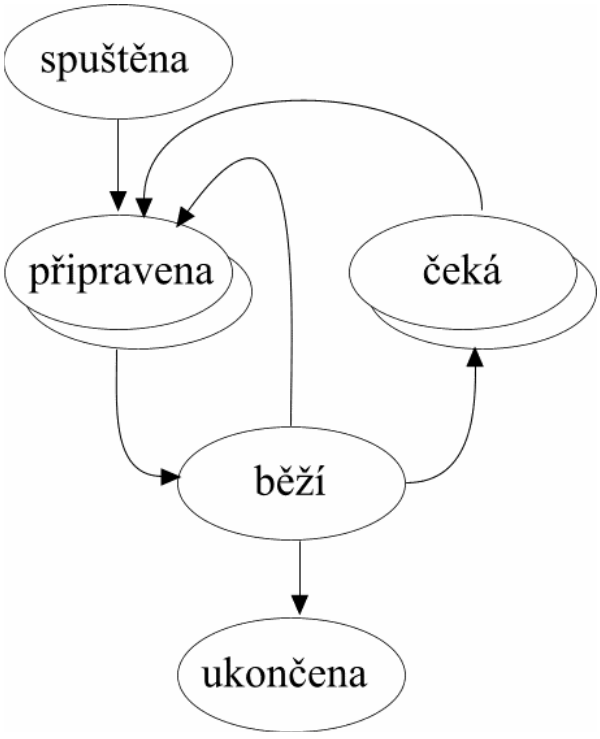
Aplikační programy stojí v **hierarchické struktuře virtuálního počítače** nejvýše a pro svou práci využívají systémové programy a jádro. Principem hierarchické struktury je, že vyšší vrstvy mohou pro svou činnost využívat služeb vrstev nižších, ale nikdy naopak !

2 Komponenty jádra

2.1 Správa procesoru

Správa procesoru se skládá ze dvou částí. Jednou je **dispečer**, který monitoruje stav procesoru(ů) a úloh. Druhou je **plánovač procesů**, řídící přidělování času procesoru jednotlivým úlohám.

2.1.1 Plánování úloh v multitaskovém OS



obr. 2 Plánování úloh

V případě současného zpracování více úloh je nutné zajistit efektivní přidělování procesorového času. Základní schéma plánování je na obr. 2. Po spuštění je úloha zařazena do fronty připravených úloh. Připravené úlohy čekají na dokončení běhu právě běžícího procesu. Pak plánovač vybere jednu úlohu a spustí ji. Algoritmy výběru úloh se řídí kritérii, kladenými na OS (efektivní využití procesoru, rychlá

odezva systému, vyřizování úloh podle priority, efektivní využití I/O zařízení). Ukončení běhu úlohy může nastat z několika příčin:

- Úloha vyčerpala maximalní čas, který ji byl přidělen k běhu.
- Úloha požádala o službu OS nebo o ukončení činnosti (**synchronní přerušení**[1]). Pokud lze požadavek vyřídit okamžitě, úloha po jeho vyřízení pokračuje v běhu. V opačném případě je zařazena mezi čekající procesy, dokud se neuvolní požadované prostředky.
- Bylo přijato **asynchronní přerušení**. Jedná se o přerušení mimo programový systém, způsobené buď vnějším zařízením nebo chybou v systému[2].

2.1.2 Proces, zobrazení procesů v OS a jejich přepínání

Proces je tvořen spuštěným **programem** a jeho **stavem**. Stav zahrnuje údaje o vývoji výpočtu v čase – ukazatel na aktuální instrukci programu, proměnné, otevřené soubory atd. Program je algoritmus, podle kterého se stav vyvíjí.

Až na první systémový proces, který je vytvořen automaticky, každý další **proces vzniká** jako synovský proces v důsledku volání jádra rodičovským procesem. Synovský a rodičovský proces znají své identifikátory, takže spolu mohou komunikovat. Synovský proces dědí od rodiče některé jeho vlastnosti, jako prioritu přístupu k procesoru, přístupová práva k souborům atd.

Zánik procesu může být vyvolán funkcí exit nebo z vnějších příčin jádrem OS.

Po spuštění procesu OS vytvoří strukturu, v které ukládá jeho identifikaci a informace o stavu – **řídící blok procesu** (PCB - process control block). Pokud je úloha odstavena, podle zde uložených dat ji bude možné opět spustit. Jsou zde registry procesoru, protože v nich jsou obsaženy důležité stavové informace – poloha ukazatele ve vykonávaném kódu (PC), stav zásobníku (SP) a deskriptory, ukazující na paměť přidělenou procesu. Musí zde být odkazy na informace o používaných souborech a periferiích. Dále zde mohou být údaje sloužící při rozhodování plánovače úloh – priorita procesu, údaj o době čekání ve frontě atd.

2.1.3 Nízkoúrovňová synchronizace procesů

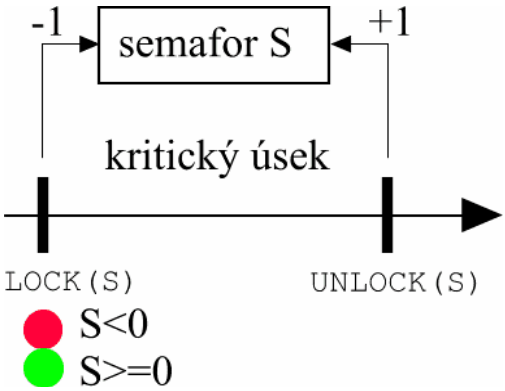
Nízkoúrovňová synchronizace procesů umožňují běh více úloh na jednom počítači. Operační systém ji využívá například při sdílení prostředků. Nutná je synchronizace i v případě víceprocesorového systému, aby bylo možné koordinovat jednotlivé větve algoritmu s různou délkou výpočtu, pokud vzájemně využívají své výsledky.

Synchronizační prostředky:

Semafor – obecný semafor hlídá úsek kódu, do kterého může vstoupit pouze omezený počet procesů (obr. 3). Semafor je přednastaven na hodnotu, která udává počet procesů, které mohou vstoupit do kritického úseku. Při vstupu se zmenší hodnota semaforu o jedna. Pokud se dostane do záporných hodnot, proces je odložen do fronty čekajících procesů. Při průchodu koncem kritického úseku se zvýší

[1] Zjednodušeně: přerušení vyvolané použitím instrukce `int` v programu

[2] Narozdíl od 1, přerušení přišlo jako elektrický signál na vstup INT procesoru



obr. 3 Semafor

semafor o jedna. Je-li v kladných hodnotách, prohlédne se fronta čekajících procesů, zda některý není odstavený na semaforu a spustí se.

Zprávy – procesy si navzájem mohou zasílat zprávy. Proces zašle zprávu pomocí funkce `SEND(Prijemce,Zprava)`. Proces `Prijemce` si zprávu může vybrat pomocí funkce `RECEIVE(Odesilatel,Zprava)`. `Odesilatel` i `Zprava` jsou výstupní parametry. Pokud žádná zpráva procesu nedošla, je pozastaven.

2.1.4 Zablokování – deadlock

Zablokování procesu je stav, kdy proces čeká na přidělení prostředků, které využívá někdo jiný. Při běhu několika procesů současně může dojít k následující situaci:

- proces A obsadí prostředek 1.
- proces B obsadí prostředek 2.
- proces A žádá o přidělení prostředku 2 a je odstaven, protože ten je blokovaný procesem B
- proces B žádá o přidělení prostředku 1 a je odstaven, protože ten je blokovaný procesem A.

V tuto chvíli se ani jeden proces již nerozeběhne, protože si navzájem překážejí v další práci. Aby se této situaci předešlo, používají se následující metody:

1. Úplné počáteční přidělení prostředků. Proces musí všechny prostředky alokovat na počátku běhu a jsou mu celou dobu přiděleny. Nevýhodou je, že uživatel nemusí všechna potřebná zařízení znát předem a prostředky se využívají velmi neefektivně.
2. Dynamické přidělování prostředků. Proces při spuštění specifikuje požadované prostředky, ale systém ho spustí i v případě, že všechny nejsou k dispozici. Během běhu se hlídá, zda není současně spuštěna jiná úloha, vyžadující stejné prostředky. V takovém případě dojde k zablokování jedné úlohy do uvolnění prostředků. Metoda není efektivní, může blokovat procesy i v případě, že to není nutné.
3. Hierarchické přidělování prostředků. Prostředky mají přidělená identifikační čísla. Pokud o ně žádá více procesů, jsou přiděleny tomu, který postupuje podle rostoucích identifikačních čísel. Technika se používá ve specializovaných systémech, u nichž je možné odhadnout posloupnost prostředků, kterou obvykle procesy budou využívat.
4. Detekce a zotavení. Systém má k dispozici tabulku *obsazených prostředků* a tabulku *čekajících procesů* na přidělení prostředku. Tabulky se obnovují při

obsazení nebo uvolnění prostředku. Pokud je požadován uzamčený prostředek, provede se algoritmus analyzující zablokování. V případě, že k deadlocku došlo, je třeba, aby jeden z procesů prostředky uvolnil (systém se musí zotavit). Zotavení používá v podstatě dvě metody:

- Speciální algoritmus pro zpětný chod výpočtu do stavu před alokací
- Kontrolní body (snapshoty), v kterých se ukládá stav procesu před alokací prostředků – do nich se provádí návrat.

2.2 Správa paměti

Sleduje stav volné paměti a které části paměti jsou kým užívány. Řídí přidělování a uvolňování paměti. Pro přidělování volné paměti existují různé strategie, z nichž některé vyžadují podporu technického vybavení počítače (tzv. MMU – memory management unit – jednotku pro správu paměti).

2.2.1 Dynamické přidělování bloků paměti

Nejjednodušším způsobem přidělování paměti je statické přidělení volné paměti procesu při jeho spuštění. Proces dostane veškerou paměť zaalokovanou při startu a již její množství nemůže dále ovlivnit. Nevýhodou je neefektivní správa paměti.

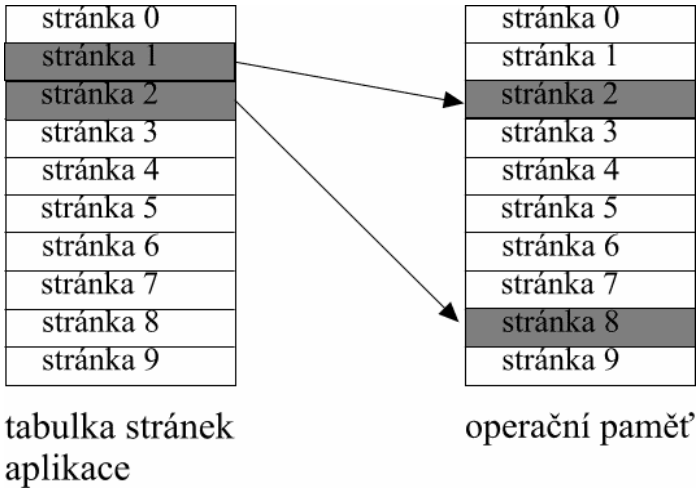
Nejčastěji se využívá **přidělování bloků** paměti **na žádost procesu**. Pokud je k dispozici volná paměť, operační systém přidělí procesu blok požadované velikosti. OS musí mít přehled o stavu paměti (vést si seznam volných a obsazených oken) a zvolit vhodnou strategii pro přidělování paměti (například alokace místa ve volném okně s nejbližší větší velikostí, než činí požadavek). Během práce systému vzniká totiž problém s **fragmentací** paměti. Procesy střídavě uvolňují a alokují bloky, takže přidělený paměťový prostor procesu nakonec nezabírá souvislý kus paměti. Postupně tak dojde k situaci, kdy systém nenajde pro přidělení dostatečně velké okno v paměti, ačkoliv součet volných bloků je větší než požadované místo.

Problém s fragmentací je možné řešit bez podpory HW **setřásáním**, kdy se v případě velké fragmentace spustí algoritmus, spojující oblasti volné paměti. Jedná se o časově náročnou záležitost, kterou komplikuje skutečnost, že se mění umístění dat v paměti. Procesy tedy musí být informovány o přeadresování. Problém je řešitelný speciálními technikami, v mezním případě se řeší tak, že se restartují všechny spuštěné procesy.

2.2.2 Virtualizace paměti

Pokud je k dispozici MMU, fragmentace se odstraňuje pomocí mechanismu stránkování. Programy využívají virtuální adresu, kterou HW pomocí tabulky stránek překládá na adresu fyzickou. Pokud aplikace vyžaduje větší blok paměti, než je k dispozici, zaalokují se do dvou po sobě jdoucích stránek virtuální paměti fyzické stránky, které spolu nemusí sousedit (obr. 4).

Stránkování umožňuje provozovat OS **virtuální paměť**. Jedná se o to, že úlohám může být přiděleno více místa, než je skutečná velikost operační paměti. Část stránek se pak odkládá na externí paměťové médium a do operační paměti se nahrává teprve v případě, že je proces vyžaduje aktuálně pro čtení nebo zápis. Mechanismus se nazývá **stránkování na žádost**. Položky v tabulce stránek obsahují příznak, určující, zda je stránka v operační paměti. Pokud ne, je vyvoláno přerušení a stránka se dotáhne z disku.



obr. 4 Stránkování paměti

Algoritmy pro výměnu stránek mezi operační a externí pamětí vyžadují v tabulce stránek další informace:

- Jak často je stránka používána / kdy byla natažena.
- Zda byl do stránky proveden zápis od posledního natažení do paměti.

Základní algoritmy výměny stránek:

FIFO (first in first out) – bude se nahrazovat stránka, která je nejdéle v paměti.

LRU (least recently used) – bude se nahrazovat nejméně užívaná stránka

Příznak zápisu do stránky od posledního natažení umožní stránku přepsat bez ukládání, pokud nebyla změněna.

2.3 Správa souborového systému

Souborový systém je část OS, která umožňuje ukládání dat převážně sekvenčním způsobem do pojmenovaných objektů - **souborů**. Souborový systém by měl být organizován tak, aby umožňoval uživateli shodnou práci s co možná největším množstvím zařízení. Některé periferie (např. tiskárna, klávesnice) mohou být namapovány do souborového systému jako tzv. **speciální soubory** a lze k nim jako k souboru přistupovat (v tomto případě jsou pouze pro zápis nebo pro čtení).

Struktura souboru obsahuje mimo vlastních dat organizační informace: Jméno, délku, vlastníka, přístupová práva, časy vzniku, změny a posledního přístupu.

OS obvykle poskytuje následující služby pro práci se soubory: otevření (create), zavření (close), smazání (remove), sekvenční čtení (read) a zápis (write), změnu pozice v souboru (seek), změnu přístupových práv (chmod) a vlastníka (chown).

OS vytváří na zařízení **adresáře** - speciální soubory, které obsahují informace o logickém umístění a názvu ostatních souborů. Struktura adresáře bývá různě složitá – lineární seznam u nejstarších OS, strom, les (několik stromů – viz disky v MS-DOSu), orientovaný graf (UNIX – možnost linků mezi adresáři).

Služby OS nad adresáři jsou: vytvoření (mkdir) a smazání (rmdir) adresáře, změna adresáře (chdir) aj.

2.4 Správa vstupních a výstupních zařízení

Úlohou vstupně/výstupního subsystému je zajistit uživatelským aplikacím shodný přístup k různým typům zařízení stejné třídy (monitory, tiskárny) – **nezávislost na zařízení**. Musí také umožnit **sdílení** mezi procesy a **obsluhovat chyby** od zařízení.

Zařízení se z hlediska přístupu dělí na dva základní druhy – na **zařízení znaková a bloková**. Znaková při jedné v/v operaci pracují vždy s jedním znakem (klávesnice – standartní vstup, terminál – standartní výstup). Bloková pracují najednou s většími objemy dat (disky – minimálně jeden sektor).

Podle typu zařízení OS musí zvolit vhodnou strategii přidělování. Některá zařízení, např. disky, může současně používat více procesů. Jsou to **sdílená** zařízení. Naopak např. tiskárnu může vlastnit vždy pouze jedna úloha, protože by se pomíchaly části různých dokumentů do jednoho výstupu. To jsou **dedikovaná** zařízení.

V/V subsystém lze hierarchicky rozčlenit do několika vrstev:

- Obslužné rutiny přerušení
- Ovladače V/V zařízení
- Části jádra nezávislé na zařízení
- Programátorské rozhraní k perifériím

3 Literatura

1. Operační systémy; S.E. Madnick, J.J. Donovan; SNTL 1983
2. Operační systémy; J. Lažanský; http://labe.felk.cvut.cz/ftp/vyuka
3. Metodiky programování; J. Lažanský; http://labe.felk.cvut.cz/ftp/vyuka
4. Vybrané partie z operačních systémů; http://linux-vos.felk.cvut.cz/vps/index.html