Operační systémy

čtvrtek 8:50 A11

Petr Kolář

místnost A-10

tel: +420-485 353 673

e-mail: `Petr.Kolar@vslib.cz`

WWW: `http://www.kai.vslib.cz/~kolar/os/`

1 Doporučená literatura

studijní text na `http://www.kai.vslib.cz/~kolar/os/`

Abraham Silberschatz, Peter B. Galvin: Operating System Concepts, 5th edition Addison Wesley 1998; ISBN 0201591138.

Andrew S. Tannenbaum: Modern Operating Systems, 2nd edition, Prentice Hall, 2001; ISBN 0130313580.

nebo libovolná kniha o **principech** operačních systémů

Obrovské množství relevantních materiálů je na Internetu.

2 Počítače a jejich historie

2.1 Druhy počítačů

(Číslicový) počítač = zařízení na zpracování informací řízené programem umístěným v paměti (dříve se používal termín samočinný počítač).

Analogový počítač = zařízení na simulování fyzikálních dějů pomocí elektrických veličin.

Hybridní počítač = analogový počítač řízený číslicovým.

Budeme se zabývat pouze číslicovými počítači.

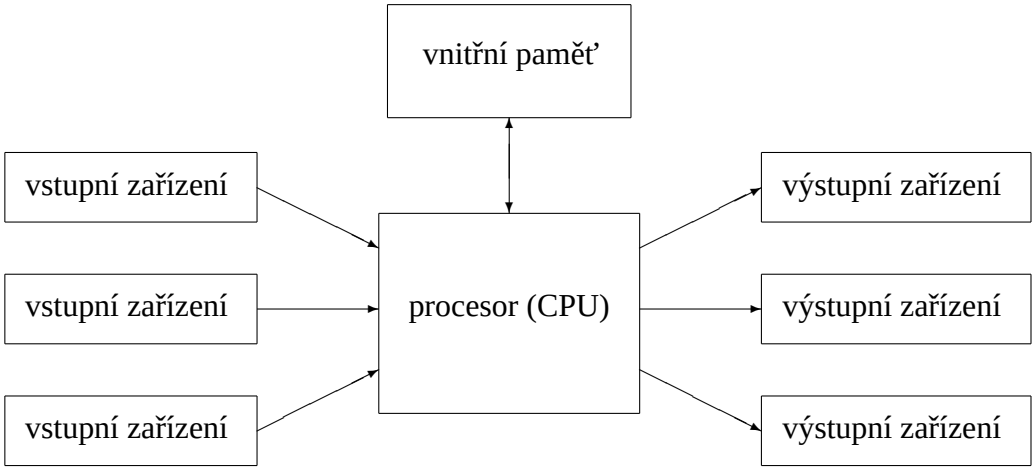
2.2 Počítačový systém

Pro používání číslicového počítače je nutné:

- technické vybavení (hardware) – vlastní počítač (obsahuje některá I/O zařízení jako disky) a další zařízení – displej, klávesnice, myš, tiskárna, . . .
- programové vybavení (software)
  - aplikační programové vybavení – umožňuje na počítači provádět nějakou užitečnou činnost (např. zpracování textů, výpočty, kreslení technických výkresů, zpracování obrazu, zvuku nebo videa)
  - systémové programové vybavení – umožňuje efektivní používání počítače
    - operační systém
    - ostatní systémové programy

2.3 Struktura počítače

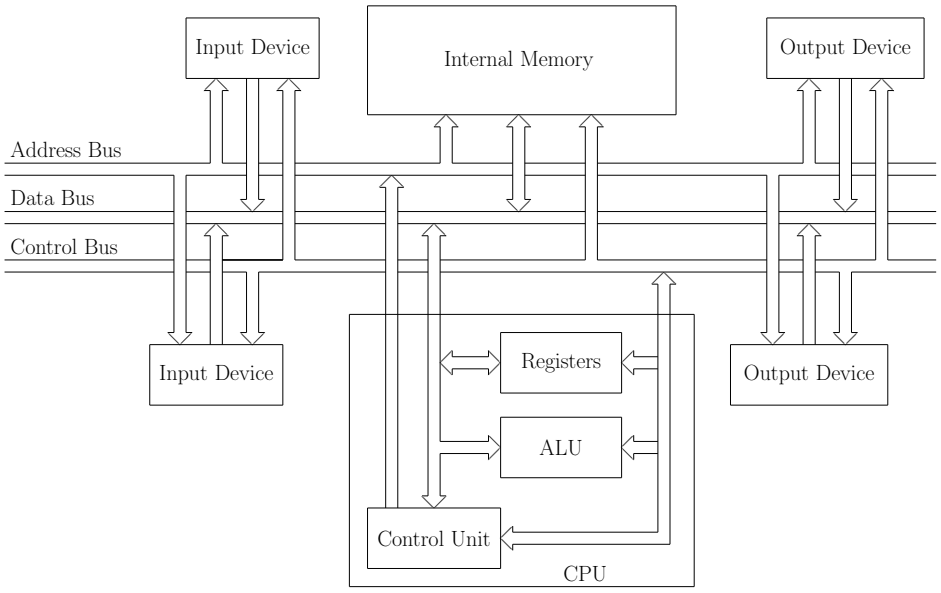
2.3.1 Von Neumannovo schéma počítače



- jeden procesor, jeden proud řízení
- vnitřní paměť typu RWM-RAM používaná pro uložení dat i programu
- vstupy a výstupy (V/V, input/output – I/O)

Program je posloupnost instrukcí uložených ve vnitřní paměti. Procesor načítá program po jednotlivých instrukcích a postupně tyto instrukce provádí.

Jednotlivé části počítače jsou propojeny pomocí sběrnice.



Procesor:

- aritmeticko-logická jednotka (ALU)
- registry
- řadič (control unit)

2.4 Paměti

RAM Random Access Memory – paměť s adresním (nebo libovolným) přístupem (častý překlad „s náhodným“ přístupem je nevhodný) – výběr požadované buňky paměti se děje pomocí čísla, adresy (podobně jako číslo domu v ulici nebo číslo pokoje v hotelu).

adresa:	0	1	2	3	4	5	6	7	8	9	10	...
obsah:	235	72	144	0	255	4	68	105	115	107	0	

RWM Read-Write Memory – paměť pro čtení i zápis – používána jako vnitřní, hlavní paměť počítačů (např. 256 MB RAM)

ROM Read-Only Memory – paměť pouze pro čtení – obsah určen při výrobě, ve starších osobních počítačích používána pro uložení ROM BIOSu (basic input/ouput system), který zajišťuje detekci a rychlé otestování hardwaru a inicializaci počítače včetně zavedení operačního systému; v novějších PC je alespoň část této paměti realizována pomocí flash ROM

PROM Programmable ROM – obsah paměti lze jednou zapsat pomocí speciálního zařízení, pak lze pouze číst

EPROM, EEPROM, EAROM – Erasable PROM – jako PROM, ale obsah lze vymazat a paměť (mnohokrát) znovu naprogramovat

flash ROM – paměť, kterou lze číst i zapisovat přímo v počítači

2.4.1 Příklady pamětí s jiným přístupem než RAM

- paměti se sekvenčním přístupem – lze číst pouze od začátku do konce, fungují podobně jako magnetofonové nebo video pásky a kazety; adresa je potřebná pouze pokud se má část obsahu přeskočit pro určení začátku
- zásobník LIFO (Last In, First Out) – též stack nebo FILO – lze ukládat libovolný počet položek; při výběru se položky čtou od poslední uložené; pro přístup není potřeba adresa
- asociativní paměť, CAM (Content Addressed Memory) – ptám se na obsah, paměť vrací, zda jej obsahuje nebo adresu, kde se v ní zadaná data nacházejí nebo vrací hodnotu spojenou se zadaným klíčem; obvykle se realizuje programově, ale například cache procesorů obsahují hardwarovou asociativní paměť

2.5 Instrukce

V paměti uložena jako posloupnost bytů. Pro usnadnění programování se neprogramuje přímo ve strojovém kódu, ale v tzv. jazyce symbolických adres (JSA, Assembly Language), který umožňuje používat pro instrukce mnemotechnické zkratky, a symbolická jména pro návěští a proměnné. Překlad programu v JSA do strojového kódu provádí Assembler.

2.5.1 Druhy instrukcí

- pro přesuny dat (mezi registry a pamětí a mezi jednotlivými registry)

- aritmetické a logické
  - podle počtu operandů jedno-, dvou-, tří-adresní instrukce (tříadresní málokdy)
  - ADD operand 2. operand je implicitní (registr akumulátor)
  - ADD výsledek, operand1, operand2 tříadresní instrukce
- skoky, volání a návraty z podprogramu
  - podmíněné dle výsledku předchozí operace pomocí flagů (stavový registr)
  - nepodmíněné
- řízení procesoru
  - např. přechod do privilegovaného stavu a zpět, zákaz a povolení přerušení

2.5.2 Jiná schémata počítačů

Hardvardské schéma počítače – oddělená paměť pro program a pro data (některé jednočipové mikropočítače).

Víceprocesorové počítače – počítače s několika CPU. Dělí se podle toho, zda mají sdílenou paměť:

- multiprocessors (multiprocesory) mají sdílenou paměť
- multicomputers (multipočítače) nemají sdílenou paměť, procesory komunikují například pomocí mechanismu zasílání zpráv

Rozdělení počítačů podle počtu instrukčních a datových proudů:

- SISD (single instruction, single data) – běžné jednoprocesorové počítače
- SIMD (single instruction, multiple data) jedna instrukce se provádí na větší množství dat (například sčítání dvou vektorů) – tzv. vektorové počítače, některé superpočítače jsou SIMD
- MISD (multiple instruction, single data) – neexistují

- MIMD (multiple instruction, multiple data) – víceprocesorové systémy; podle toho zda mají nebo nemají sdílenou paměť se rozdělují na multiprocesory (se sdílenou pamětí) a multipočítače (multicomputers – spolupracující počítače propojené sítí)

Masivní multiprocessing – minimálně desítky procesorů.

Současné počítače jdou v mnoha detailech mimo rámec von Neumannova schématu – existují DMA kanály a bus mastering karty, které řídí přenosy dat mezi pamětí a I/O zařízeními bez účasti procesoru. Některá PC mají více procesorů. Moderní procesory zpracovávají několik instrukcí současně. Většina procesorů Intel Pentium (kromě nejstarších) obsahuje instrukce SIMD.

2.6 Vývoj počítačů

2.6.1 První generace (1945 až 1955)

První počítače reléové a elektronkové. Obrovský příkon, velká poruchovost, velmi vysoká cena, malá rychlost. Program se psal přímo ve strojovém kódu (neexistovaly ani Assemblery). Zpočátku se sestavoval na propojovacích deskách, později se zaváděl z děrné pásky nebo děrných štítků. Výstup na řádkovou tiskárnu nebo na děrovač štítků nebo děrné pásky. Ovládání počítače z konzole. Jeden tým lidí, kteří pracovali jako konstruktéři, programátoři, operátoři i technici. Úspěchem bylo, když během výpočtu nedošlo k poruše. Použití pro numerické výpočty. Operační systémy ani programovací jazyky neexistovaly.

2.6.2 Druhá generace (1955 až 1965)

Tranzistorové počítače. Zlepšení všech parametrů, začátek obchodu s počítači. Stále vysoká cena → snaha o co nejvyšší využití počítače → dávkové systémy (batch systems). Dávka tvořená jednou úlohou (job), která se skládá z jednoho nebo několika kroků (steps), se zavádí do počítače z děrných pásek, štítků nebo z magnetické pásky. Výstup na magnetickou pásku, z ní se nespřaženě (off line) na menším specializovaném počítači prováděl výstup na tiskárnu nebo děrovač. Programy psané v jazyce symbolických adres, ve Fortranu, Cobolu, Algolu. Prodej strojového času. Operační systémy složené z několika částí:

- obsluha vstupních/výstupních zařízení

- plánovač úloh řízený jazykem pro řízení úloh (job control language)

2.6.3 Třetí generace (1965 až 1980)

Počítače s integrovanými obvody. Pozorování: výkon počítače je úměrný druhé mocnině jeho ceny → nákup toho nejsilnějšího počítače, jaký je možné si dovolit; prodej strojového času, další snahy o co nejlepší využití počítače. Velké střediskové počítače – mainframes. Na druhé straně minipočítače. První mikropočítače. Nejvíce zdržují I/O operace → multiprogramování: dokud jeden program čeká na provedení I/O operace, druhý je zpracováván procesorem. Proces – vykonávaný program, je dynamický (obsahuje programy a data, která se mohou časem měnit). Multitasking – pokud není procesu odebrán procesor protože nezačal provádět I/O operaci, je mu procesor odebrán po uplynutí jistého časového intervalu (časové kvantum) a přidělen jinému procesu. Procesy se střídají tak rychle, že je možný i interaktivní přístup. Některé systémy umožňují, aby uživatel spustil současně několik procesů, které spolu mohou komunikovat.

2.6.4 Čtvrtá generace (od roku 1980)

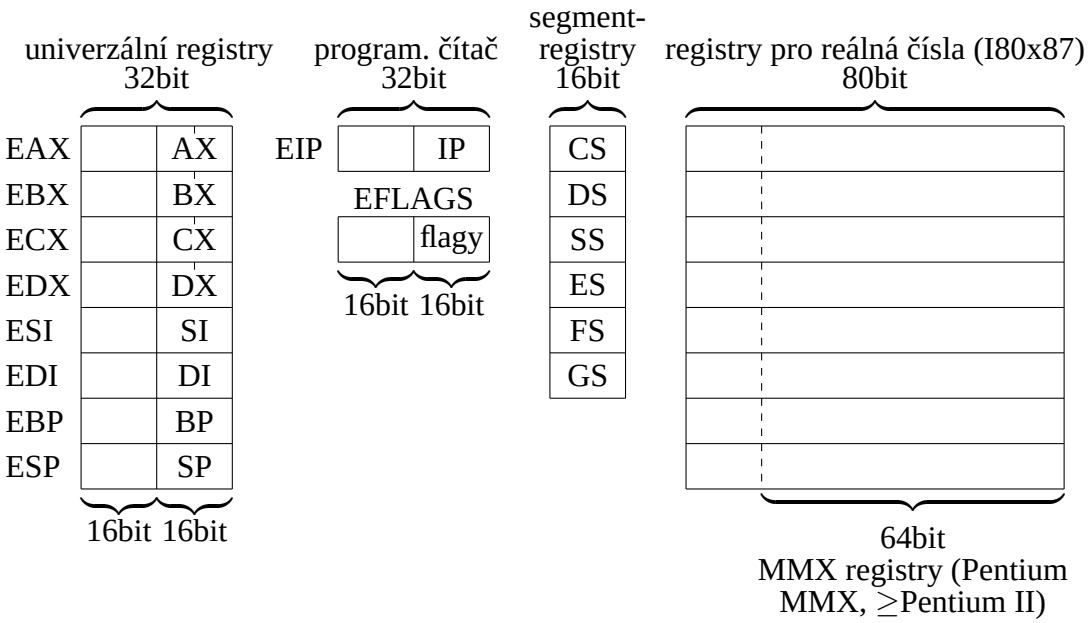
Osobní počítače s mikroprocesory. Pracovní stanice. Nárůst spolehlivosti, pokles ceny → ústup od střediskových počítačů – neplatí pravidlo ceny z minulé generace. Zvláštní fenomén – osobní počítače PC a OS MS-DOS. Grafická uživatelská rozhraní. Sítě. Síťové operační systémy (počítače propojené sítí), distribuované systémy (celá síť se chová jako jediný výpočetní prostředek). Multiprocesorové systémy, síťové OS. MS-DOS (živá zkamenělina).

2.7 Vývoj mikroprocesorů

- první pro terminál kolem 1973 Intel 4004 (čtyřbitový)
- 8008 (osmibitový, 16kB, zásobník hloubky 8), nebyl schopen při přerušení uložit stav procesoru
- 8080 používaný předchůdci dnešních osobních počítačů, OS CP/M
- Z80 jedno napájení, méně okolních obvodů, těžil z nemožnosti patentování instrukční sady
- Motorola 6800

- kolem roku 1977 Intel 8086 (šestnáctibitový)
- Motorola 68000
- 8088 – levnější verze 8086
- 80186, 286, 386, 486, Pentium → PC
- Motorola 68010, 20, 30, 40, 60 → MAC, Amiga
- Motorola byla dříve lepší než Intel, díky IBM neměla šanci
- RISC – redukovaný instrukční soubor

2.8 Sada registrů 16 a 32 bitových procesorů Intel



U procesorů před typem 80386 nebyly registry EAX, . . . EIP, EFLAGS, ale pouze šestnáctibitové AX, BX, CX, DX, SI, DI, BP, SP, IP a příznaky (flags); segmentové registry byly pouze 4 – CS, DS, SS a ES.

Každý z šestnáctibitových registrů AX, BX, CX a DX může být používán jako dvojice osmibitových registrů: AX=AH+AL, BX=BH+BL, CX=CH+CL, DX=DH+DL. Přitom xH je horních 8 bitů, xL spodních 8 bitů.

Registry pro výpočty v pohyblivé řádové čárce jsou součástí numerického koprocesoru (který byl instalován pouze do malé části osobních počítačů). U procesorů 486 DX a Pentium je numerický koprocesor součástí procesoru.

Jako MMX registry se používají registry numerického koprocesoru. Mohou být použity pro výpočty s následující typy dat:

- 8 8 bitových celých čísel
- 4 16 bitová celá čísla
- 2 32 bitová celá čísla
- 1 64 bitové celé číslo

Procesory Pentium III a novější mají zvláštní sadu osmi 128 bitových SSE registrů (Streamed SIMD EXtension), které umožňují současnou práci se čtyřmi sadami 32 bitových čísel v pohyblivé řádové čárce.

Procesory Pentium IV mohou používat SSE registry dalšími způsoby (SSE2):

- 4 32 bitové čísla v pohyblivé řádové čárce (původní SSE)
- 2 64 bitové čísla v pohyblivé řádové čárce
- 16 8 bitových celých čísel
- 8 16 bitových celých čísel
- 4 32 bitová celá čísla
- 2 64 bitová celá čísla
- 1 128 bitové celé číslo

3 Operační systémy

3.1 Funkce operačního systému

- správce prostředků – resource manager
- virtuální počítač – virtual machine

Správce prostředků. Prostředky jsou I/O zařízení, soubory, procesor, paměť apod. Operační systém vlastní jednotlivé systémové prostředky – přiděluje a odebírá je jednotlivým procesům.

Virtuální počítač. Operační systém skrývá detaily ovládání jednotlivých zařízení (transparentnost), definuje standardní rozhraní pro volání systémových služeb.

Programátor se může věnovat vlastní úloze a nemusí znovu programovat I/O operace. Program může díky „odizolování“ od konkrétních zařízení pracovat i se zařízeními, o kterých jeho autor v době vytváření programu neměl ani ponětí (program se o ovládání I/O nestará).

3.2 Rozdělení operačních systémů

Systém	Jednouživatelský	Víceuživatelský
Jednoúlohový	MS-DOS, CP/M	(stanice v Novellu), Intellec SIV
Víceúlohový	Windows, MacOS	Unix, VM/S

Charakteristickým rysem víceuživatelského OS je existence nástrojů pro omezování práv jednotlivých uživatelů. Díky tomu nemůže obyčejný uživatel mazat systémové soubory ani soubory jiných uživatelů, nemůže násilím ukončovat běh systémových procesů a procesů jiných uživatelů, apod.

Existence uživatelských profilů, které umožňují uživateli měnit některé rysy vzhledu a chování OS a programů, není dostatečným důvodem pro to, aby byl OS považován za víceuživatelský.

Další rozdělení OS podle způsobu nasazení:

- dávkový OS
- interaktivní OS
- operační systém reálného času

Rozdělení OS pro víceprocesorové stroje

- Asymmetric multiprocessing – pouze jeden procesor smí pracovat se systémovými datovými strukturami. Výhody: jednodušší – není potřeba, aby OS umožňoval sdílení svých vnitřních datových struktur. Nevýhody: nižší pružnost, v některých případech nižší výkonnost.
- Symmetric multiprocessing – se systémovými datovými strukturami může pracovat více procesorů.

3.3 Hardwarové prostředky využívané operačním systémem

3.3.1 Přerušení (interrupt)

Původně mechanismus, kterým si řadiče (například I/O zařízení) mohou vyžádat pozornost procesoru. Nyní používáno i k dalším účelům.

Při vyvolání přerušení procesor začne provádět podprogram obsluhy přerušení (interrupt service routine – ISR) podobným způsobem, jako by byl vyvolán normální podprogram. Podprogram musí uchovat stav procesoru, pak provede vlastní obsluhu přerušení (například zašle znak nebo blok znaků na výstupní zařízení) a nakonec obnoví stav procesoru, aby přerušený program nic nepoznal (až na zpoždění). Podobá se vyvolání podprogramu, ale provádí se speciální instrukcí.

Rozdělení přerušení:

- vnější – zdrojem jsou řadiče (zejména I/O zařízení) umístěné „vně procesoru“. K přerušení dochází bez ohledu na právě prováděné místo v programu a ISR je vyvolán po dokončení instrukce. Reakci na přerušení lze dočasně zakázat (maskovat), pak k obsluze dojde po povolení přerušení. Po návratu z ISR přerušený program pokračuje další instrukcí.
- vnitřní – přerušení je vyvoláno chybou při provádění strojové instrukce (dělení nulou, přetečení, porušení ochrany paměti, výpadek stránky). ISR může vypsat chybové hlášení a ukončit program, dosadit náhradní výsledek v případě aritmetické chyby, zavést stránku do vnitřní paměti z disku apod.. Při některých chybách je možné zopakovat instrukci, která chybu způsobila.
- programové – přerušení je vyvoláno instrukcí volání přerušení umístěnou přímo v programu. Používá se pro volání služeb operačního systému. Výhoda oproti volání podprogramů: není možné vyvolávat podprogramy na libovolných adresách (pouze adresy uvedené v tabulce přerušení).

3.3.2 Vstupní a výstupní řadiče

Slouží k připojování I/O zařízení. Z hlediska programátora řadič vypadá jako sada hardwarových registrů. Registry:

- jen pro čtení
- jen pro zápis

- pro čtení i zápis

Při inicializaci počítače je potřeba zjistit, které řadiče jsou v počítači zapojeny a inicializovat je. Při vstupu nebo výstupu dat je zpravidla nutné čtením z řadiče zjistit, zda je zařízení připraveno, zapsat do řadiče příkaz, a zapsat nebo přečíst data. Pokud není možné ihned pokračovat, zařízení zpravidla signalizuje svoji připravenost vyvoláním přerušení.

Podle architektury počítače se vstupy/výstupy dělí na

- paměťově mapované: registry jsou adresovány jako paměť, přístupné pomocí běžných operací čtení a zápisu do paměti
- izolované: registry jsou přístupné pomocí speciálních instrukcí (zpravidla nazývaných IN a OUT); díky tomu jsou adresní prostory paměti a vstupů/výstupů oddělené

3.3.3 Kanály

Pro odlehčení procesoru bývají součástí počítače obvody schopné realizovat větší množství I/O operací (jedná se o odchylku od Von Neumannova schématu počítače):

- DMA kanály. Pro kopírování bloků dat mezi pamětí a I/O zařízením. Je třeba je naprogramovat zápisem do hardwarových registrů.
- Specializované I/O procesory (někdy nazývané kanály). Jsou řízeny posloupností vlastních instrukcí (tzv. kanálovým programem).
  - selektorové – obsluhuje 1 rychlé zařízení (mg. pásky)
  - multiplexní – mohou obsluhovat několik pomalých zařízení (tiskárny, někt. terminály, děr. pásky)

3.3.4 Privilegované instrukce

Některé instrukce by při nevhodném použití mohly vést ke zhroucení operačního systému, k chybám, nebo k narušení bezpečnosti. Jedná se o I/O instrukce a některé instrukce řízení procesoru. Aby se znemožnilo jejich použití uživatelskými programy, má většina procesorů dva nebo více stavů s různým stupněm

privilegií. Příklad dva stavy: privilegovaný a uživatelský. Po startu je procesor v privilegovaném stavu a je spuštěn OS. V privilegovaném stavu je dovoleno provádět i privilegované instrukce, běží v něm operační systém. V uživatelském stavu pokus o provedení privilegované instrukce skončí chybou, používají jej uživatelské programy. Při vyvolání služeb operačního systému přechází procesor do privilegovaného stavu, při návratu ze systémové služby přechází do uživatelského stavu.

3.4 Hardwarové prostředky počítačů PC

3.4.1 Porty

Porty jsou základní prostředek pro komunikaci počítače s přídavnými zařízeními. Slouží k obousměrné komunikaci. Zápis na port se používá pro výstup dat nebo příkazů na zařízení, čtení z portu slouží ke čtení dat a stavu zařízení. Porty se rozlišují číslem – adresou portu. U procesorů řady 80x86 je k dispozici 64 K (tj 65536) portů, na většině počítačů PC kompatibilních je však dostupných pouze 1024 portů na adresách 0 až 1023 tj. 0 až 3FFh. Většina zařízení obsazuje několik portů, často 8 nebo 16 (vždy mocnina čísla 2). Problémem bývá, že u některých přídavných zařízení je v dokumentaci sice uvedeno nejnižší číslo portu, ale chybí údaj o tom kolik portů zařízení obsazuje. U zařízení, které umožňují výběr několika nastavení je možné zhruba tento údaj odhadnout pomocí rozestupů jednotlivých nastavení. Například lze-li nastavit síťovou kartu na adresy 300h, 320h a 340h, je pravděpodobné, že používá 20h = 32 portů.

3.4.2 Přerušení

Přerušení umožňuje jednotlivým zařízením signalizovat procesoru, že vyžadují obsluhu. Díky přerušením nemusí počítač jednotlivá zařízení periodicky testovat. Pokud zařízení vyvolá přerušení, počítač provede podprogram pro obsluhu tohoto přerušení. Tento podprogram zjistí příčinu a provede požadované akce. Na počítači bývá aspoň jedno přerušení, které se vyvolává periodicky a slouží k počítání času. V obsluze tohoto přerušení mohou být obsloužena i zařízení, které nevyžadují častou a pohotovou obsluhu (například tiskárna) a která sama vlastní přerušení nepotřebují. Starší počítače PC a PC/XT mají 8 přerušení (0-7). Šestnáctibitové karty v počítačích PC/AT a novějších mohou používat dalších 8 přerušení (8-15, tj, 8 až 0Fh).

Počítače PC, PC/XT

IRQ	Zařízení
0	Časovač
1	Klávesnice
2	EGA, VGA
3	COM2, (COM4)
4	COM1, (COM3)
5	Řadič hard disku (XT)
6	Řadič floppy disku
7	LPT1

Počítače PC/AT a lepší

IRQ	Zařízení
0	Časovač
1	Klávesnice
2	Kaskáda IRQ 8 až IRQ 15
3	COM2, (COM4)
4	COM1, (COM3)
5	(LPT2)
6	Floppy disk
7	LPT1
8	Hodiny, kalendář, budík
9	(VGA)
10	(Zvuková karta)
11	
12	(PS/2 Myš)
13	(Numerický koprocesor 80x87)
14	Řadič IDE Hard Disku
15	(Řadič IDE Hard Disku 2)

3.4.3 Paměť

Pro přenos bloků dat mezi zařízením a počítačem se někdy používají speciální mechanismy. Jedním z nich je sdílená paměť. Jedná se o paměť RAM, která se objeví někde v adresním prostoru procesoru. Do této paměti lze zapisovat bloky dat určené k výstupu na zařízení a naopak bloky dat přečtené ze zařízení lze z této paměti číst.

3.4.4 DMA kanály

Další metodou pro přenos bloků dat mezi počítačem a zařízením je DMA (direct memory access – přímý přístup k paměti), který umožňuje přenos bloků dat mezi portem a pamětí bez účasti procesoru, často i souběžně s jeho další činností. Všechny počítače PC mají 4 osmibitové DMA kanály (0-3), počítače PC/AT a novější mají navíc 4 šestnáctibitové DMA kanály (4-7).

Počítače PC, PC/XT

DMA	Zařízení
0	
1	
2	Řadič pevného disku (XT)
3	Řadič floppy disků

Počítače PC/AT a lepší

DMA	Zařízení
0	
1	
2	Řadič floppy disků
3	(Enhanced LPTx)
4	Kaskáda DMA 0 až DMA 3
5	
6	
7	Řadič IDE disků na ISA

4 Zavádění operačního systému do počítače

Při startu nebo resetu počítače PC se spustí program uložený v paměti ROM. Tato paměť obsahuje podprogramy pro práci s jednotlivými vstupními a výstupními zařízeními, Basic Input-Output System - BIOS, proto se nazývá ROM BIOS. BIOS podprogramy byly používány OS MS-DOS, moderní operační systémy jako jsou Windows nebo různé varianty Unixu jich používají minimum. ROM BIOS obsahuje tak zvaný POST power-on self-test, který slouží k detekci, otestování a inicializaci jednotlivých částí počítače. Po skončení POST se BIOS pokusí zavést operační systém, u starých počítačů nejprve z diskety v první disketové mechanice a pak z prvního pevného disku, u novějších se pořadí mechanik zadává v konfiguraci BIOSu. U počítačů vybavených síťovou kartou s Boot ROM může být řízení předáno programu v této Boot ROM, která může zavést operační systém

ze síťového serveru nebo pokračovat v zavádění systému z lokálních disků.

Při zavádění systému z lokálního disku se postup poněkud liší podle toho, zda se systém zavádí z pevného disku nebo z diskety. Zatímco diskety mohou obsahovat pouze jeden systém souborů, pevné disky mohou být rozděleny na části, oddíly (partitions). U ostatních médií je situace individuální - některé USB disky se chovají jako diskety, jiné jako pevné disky, ZIP média se chovají jako pevné disky, ale v BIOSu některých počítačů lze nastavit mapování, při kterém je vidět pouze obsah jednoho oddílu.

Pevné disky mají na svém začátku tak zvaný master boot record MBR. Master boot sektor obsahuje krátký (maximálně 446 bytů) zaváděcí program a tabulku oddílů (partition table). Tabulka může obsahovat informace o nejvýše čtyřech oddílech, další oddíly lze vytvářet rozdělením jednoho z nich (v terminologii používané v OS MS-DOS a Windows se takovému oddílu říká extended partition a jeho částem logical drives).

Na jednom počítači (i na jednom disku) tak může být nainstalováno více operačních systémů. Mnoho uživatelů tak má na svém počítači nainstalovány například Windows a Linux. Zavaděč umožní uživateli vybrat, jaký systém chce do počítače zavést, nebo jen z tabulky oddílů vybere oddíl, který je označen jako aktivní. Z tohoto oddílu načte úplně první sektor (zvaný System Boot Record) a předá řízení programu v tomto sektoru obsaženém.

Zaváděcí program ze System Boot Recordu postupně načte jednotlivé části operačního systému a předá systému řízení.

V operačním systému MS-DOS zavaděč načte soubor `IO.SYS`, který načte `MSDOS.SYS` (v PC-DOS se tyto soubory jmenují `IBMBIO.COM` a `IBMDOS.COM`). Podle obsahu `CONFIG.SYS` se zavedou ovladače zařízení a interpretuje se `AUTOEXEC.BAT` nebo se jenom předá řízení interpretu příkazů `COMMAND.COM`.

V systémech Windows-95 a Windows-98 se načte soubor `IO.SYS`, který podle obsahu `CONFIG.SYS` zobrazí případné menu a zavede ovladače a podle řádku `BootGUI=` v `MSDOS.SYS` buď zavede Windows (pomocí `WIN.COM`) nebo spustí `COMMAND.COM`.

V systémech Windows NT, Windows 2000 a Windows XP zavaděč načítá soubor ntldr z kořenového adresáře příslušného oddílu. Ntldr přepne do rozšířeného režimu, načte boot.ini a zobrazí boot menu (pokud je). Při výběru Windows XP ntldr spustí ntdetect.com pro detekci hardware. V kořenovém adresáři může být ještě bootsect.dos (pro dual booting) a ntbootdd.sys (pro některé SCSI adaptéry). Pak ntldr spustí `%SystemRoot%\system32\ntoskrnl.exe`

a `%SystemRoot%\system32\hal.dll`. Ještě načte registry, vybere HW profil, řídicí sadu a načte ovladače zařízení. ntoskrnl.exe spustí winlogon.exe a ten lsass.exe (Local Security Administration), který umožní uživateli zadat jméno a heslo a přihlásit se.

Při zavádění OS Linux se načte jádro OS (obvykle soubor /boot/vmlinuz*) a případně RAM disk (/boot/initrd*). Jádro inicializuje část hardwaru, připojí kořenový oddíl a spustí program init, který zajistí start dalších programů potřebných pro běh systému včetně připojení dalších oddílů a inicializace zbývajícího hardwaru.

5 Správa procesů

5.1 Programy a procesy

Program = zápis algoritmu v nějakém programovacím jazyce (například ve strojovém kódu). Je statický, neměnný (neuvažujeme-li vývoj nových verzí programů).

Proces (process, task) = běžící program. Proces je tvořen neměnným kódem programu a konstantami a proměnnými daty jako je stav procesoru, data na zásobníku, globální proměnné, halda, soubory atd.

Pro běh procesu jsou nutné následující prostředky systému:

- procesor
- vnitřní paměť
- další prostředky (I/O zařízení, soubory apod.)

5.2 Pseudoparalelismus

Cíl: Operační systém by měl umožnit uživatelům spustit několik programů současně a přepínat mezi nimi. Případně i umožnit, aby tyto programy běžely současně.

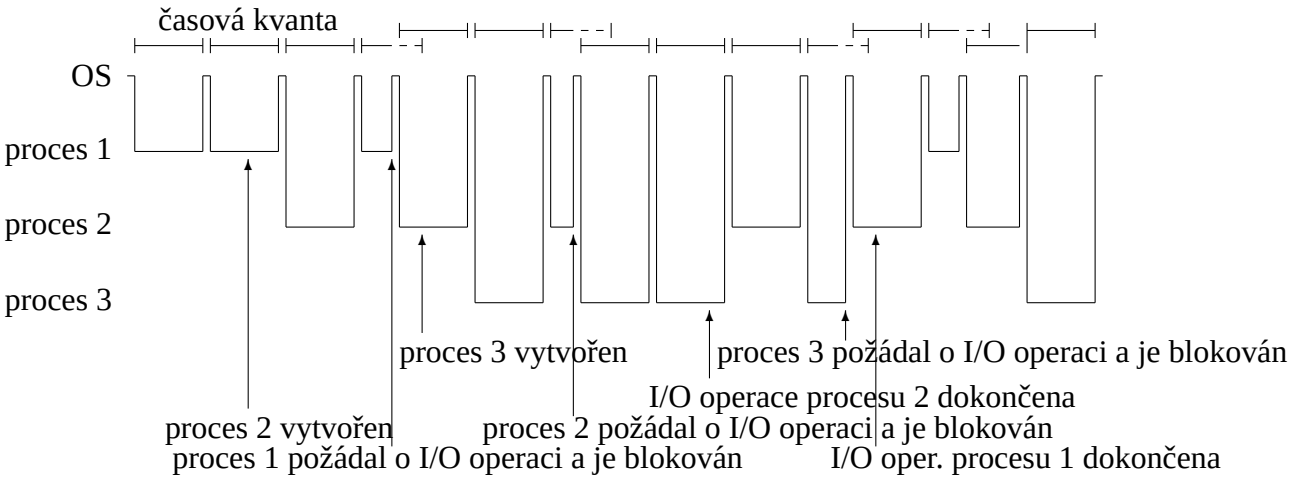
Fakt: Moderní počítače v některých ohledech porušují von-Neumann schéma počítače. Jsou **schopné současně** provádět **program** a **vstupní nebo výstupní operace**. Ale obvykle mají **pouze jeden procesor**.

Řešení: OS umožní uživateli spustit více programů. Každý program bude umístěn v jiné části paměti. OS přepíná rychle mezi jednotlivými procesy takže se zdá, že všechny procesy běží současně.

Pokud program požádá OS o I/O operaci, kterou není možné ihned dokončit, OS mu odebere procesor a přidělí jej jinému procesu.

Tento způsob práce nazýváme **pseudoparalelismus** (protože procesy ve skutečnosti neběží současně – na to by bylo potřeba více procesorů).

5.3 Diagram využití procesoru ve víceúlohovém systému

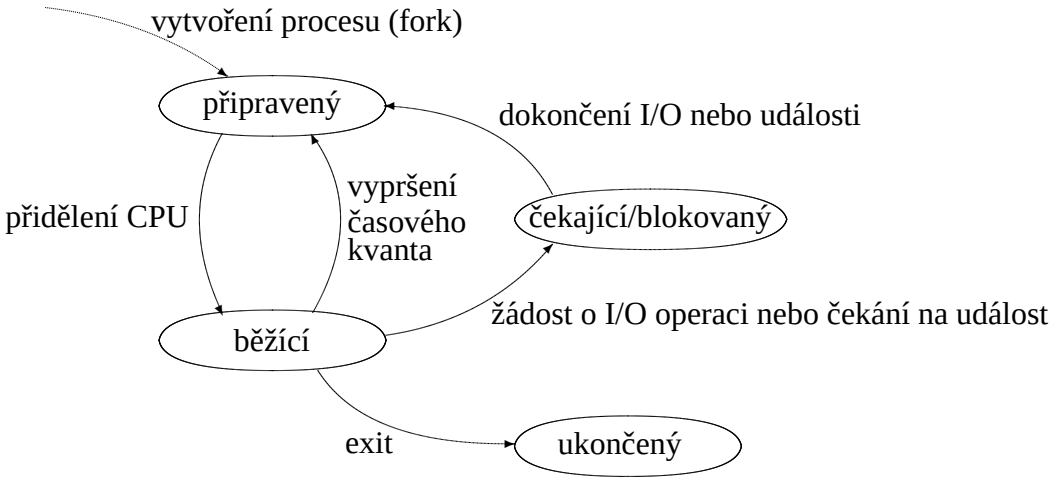


5.4 Stavy procesů

Proces je ve stavu **běžící**, pokud je mu přidělen procesor.

Proces je ve stavu **připravený**, pokud je připraven k běhu, ale není mu přidělen procesor.

Proces je ve stavu **čekající, spící nebo blokovaný**, pokud čeká na určitou událost (např. na dokončení I/O operace).



5.5 Druhy plánování procesů

Rozeznáváme tři druhy plánování procesů (process scheduling, čti šedjúling, americká výslovnost skedžling):

- krátkodobé (short-term), CPU scheduling (plánování procesoru): výběr kterému z připravených procesů bude přidělen procesor, ve všech víceúlohových systémech
- střednědobé (medium-term): výběr který blokovaný nebo připravený proces bude odsunut z vnitřní paměti na disk, je-li vnitřní paměti nedostatek (swap out, roll out)
- dlouhodobé (long-term), job scheduling (plánování prací, úloh): výběr, která úloha bude spuštěna (má význam zejména při dávkovém zpracování). Účelem je namixovat úlohy tak, aby byl počítač co nejvíce vytížen (třídy úloh podle náročnosti).

V jednotlivých OS se nemusí nutně používat všechny tři druhy plánování procesů. V některých případech je například plánování úloh zjednodušeno na pravidlo: pokud je dostatek prostředků OS, spusť proces.

5.6 Plánování procesoru

Plánování procesoru se používá ve víceúlohových systémech. Může nastat v následujících situacích:

- pokud některý proces přejde ze stavu běžící do stavu blokovaný (čekání na I/O operaci, semafor, čekání na uplynutí zadaného časového intervalu, čekání na ukončení procesu-potomka)
- pokud některý proces skončí
- pokud je některý proces převeden ze stavu běžící do stavu připravený
- pokud některý proces přejde ze stavu čekající do stavu připravený

Jestliže k plánování procesoru dochází pouze v prvních dvou výše uvedených případech, říkáme, že OS používá nepreemptivní plánování CPU (procesů). Jinak říkáme, že OS používá preemptivní plánování CPU. Přeplánování procesoru v posledním uvedeném případě se používá zřídka (například u systémů reálného času).

5.6.1 Nepreemptivní plánování

V případě nepreemptivního plánování se proces musí procesoru sám vzdát. Pokud má být doba, po kterou je proces ve stavu běžící, omezená, je nutné, aby proces kontroloval časovač a po překročení stanovené doby se dobrovolně vzdal procesoru vyvoláním služby OS, která je k tomuto účelu určena. Výhodou je, že proces nemůže být přerušen, pokud nechce (například v kritické sekci viz dále). Nevýhodou je, že špatně chovající se proces může zablokovat celý OS. Takto fungují například MS-Windows.

5.6.2 Preemptivní plánování

V případě preemptivního plánování OS může odebrat procesu procesor. Zpravidla se tak děje při uplynutí časového kvanta určeného pro běh procesu a celá akce je vyvolána přerušením od časovače. Příkladem OS, který používá preemptivní plánování je OS Unix.

5.7 Strategie plánování procesoru

Strategie použitá pro výběr, kterému z připravených procesů bude přidělen procesor, bývá tvůrci operačního systému vybírána podle těchto kritérií:

- spravedlnost: každý proces dostane spravedlivý díl času procesoru
- efektivita: udržovat maximální vytížení procesoru, příp. jiné části systému
- čas odezvy: minimalizovat dobu odezvy pro interaktivní uživatele
- doba obrátky: minimalizovat dobu zpracování každé dávkové úlohy
- průchodnost: maximalizovat množství úloh zpracovaných za jednotku času

Podle toho, které z těchto vlastností brali tvůrci systému v úvahu a jakou váhu jim přikládali, používají různé operační systému různé strategie plánování procesoru:

5.7.1 Strategie FCFS

FCFS (first come, first served – kdo dřív přijde, ten dřív mele): procesy přicházející do stavu připravený jsou umísťovány na konec fronty typu FIFO (first in first out). Při plánování procesoru se procesor přidělí tomu procesu, který je ve frontě první. Tuto strategii je možné používat při preemptivním i nepreemptivním plánování procesoru. Při preemptivním plánování se někdy nazývá round robin scheduling.

5.7.2 Strategie SJF

SJF (shortest job first – nejkratší úloha první): přednost mají úlohy, u kterých se předpokládá, že poběží krátkou dobu, nebo že využijí pouze část časového kvanta. Předpověď se provádí obvykle na základě chování úlohy v minulých časových kvantech nebo při minulých spuštěních.

5.7.3 Prioritní strategie

Prioritní strategie: každému procesu je přidělena priorita a procesy jsou ze stavu připravený vybírány podle priority. Procesy se stejnou prioritou jsou obvykle vybírány v pořadí, v jakém do stavu připravený přišly.

Problémem této strategie je, že procesy s nízkou prioritou mohou čekat neomezeně – tzv. **stárnutí – starvation**.

Stárnutí se může omezit postupným zvyšováním priority procesů, které jsou dlouhou dobu ve stavu čekající – tzv. **aging**.

5.7.4 Strategie založené na proměnné délce časového kvanta

Modifikace FCFS strategie – procesy, které mají běžet rychleji, dostanou delší časové kvantum.

Existují různé varianty, které navíc upravují délku časového kvanta podle využití předchozích časových kvant, nebo podle celkového času dosud spotřebovaného procesem.

5.8 Process Control Block

Process control block (PCB) je datová struktura, která slouží OS k práci s procesem. Jeho obsah je v různých OS různý. Může obsahovat například tyto informace:

- Informace pro správu procesů
  - CPU registry (včetně programového čítače, stavového slova programu PSW, příznaky, stack pointer)
  - stav procesu
  - ukazatel na další proces ve frontě
  - ukazatel na seznam procesů-potomků
  - ukazatel na rodičovský proces
  - skupina procesů
  - skutečné číslo uživatele (real user id)
  - skutečné číslo skupiny (real group id)
  - čas do příštího alarmu
  - ukazatel do fronty zpráv
  - příznaky nevyřízených signálů
  - číslo procesu
  - různé příznaky
- Informace pro správu procesoru
  - priorita procesu
  - časové kvantum
  - využití předešlých časových kvant
- Informace pro správu paměti

  - ukazatel na segment programu
  - ukazatele na další segmenty paměti
  - tabulky stránek
  - informace pro ochranu paměti
  - efektivní číslo uživatele (effective user id)
  - efektivní číslo skupiny (effective group id)
  - kód ukončení úlohy
  - stav signálů
- Informace pro správu souborů
  - aktuální adresář
  - kořenový adresář
  - maska práv souborů
  - popisovače souborů
  - efektivní číslo uživatele (effective user id)
  - efektivní číslo skupiny (effective group id)
- Účtovací informace
  - čas startu procesu
  - spotřebovaný čas procesoru
  - čas procesoru spotřebovaný procesy-potomky
  - počet přečetných a zapsaných diskových bloků
  - počet vytisknutých stránek na tiskárně

5.9 Změna kontextu – Context switch

U operačních systémů s preemptivním plánováním procesoru může být proces přerušen mezi libovolnými dvěma strojovými instrukcemi v programu a řízení může být předáno jinému procesu. Programy jsou však psány tak, že nepředpokládají, že by došlo ke změně obsahu registrů procesoru případně některých dalších oblastí mezi dvěma instrukcemi. Při přepnutí na jiný proces musí být také změněny další registry (ukazatel na tabulku stránek, klíč pro ochranu paměti, registr udávající, zda je procesor v privilegovaném stavu apod.).

Proto se při přepínání mezi procesy provádí tzv. uložení kontextu (context save) původně běžícího procesu a obnovení kontextu (context restore) procesu, kterému se přiděluje procesor. Pod pojmem context je myšlen stav procesoru (obsah

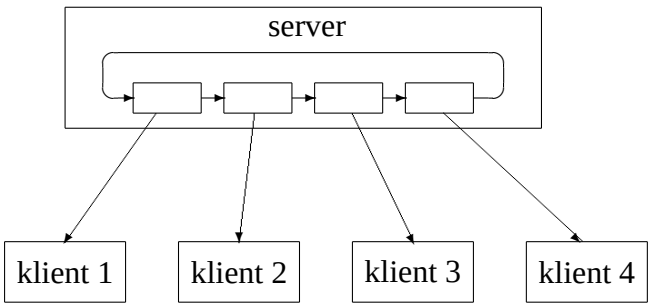
registrů), stav případného koprocesoru, případně i stav dalších zařízení. Tento context se ukládá buď na zásobník procesu, nebo do předem připravené oblasti dat v adresním prostoru procesu.

5.10 Thready

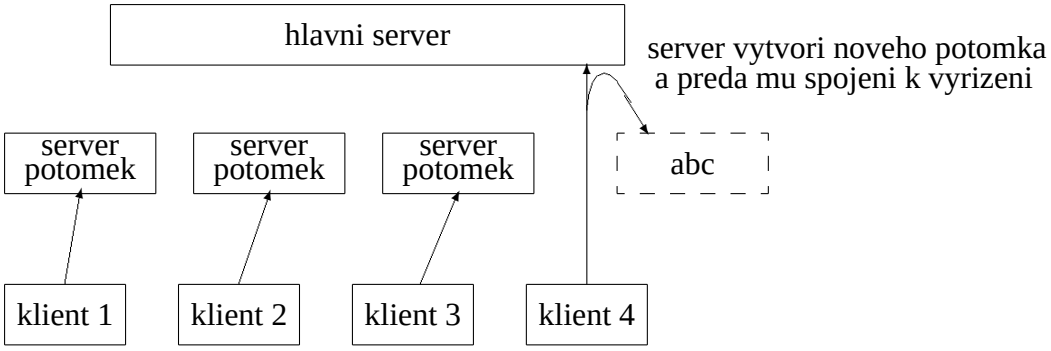
Víceúlohový systém zpravidla vytváří iluzi, že jednotlivé procesy mají celý systém pro sebe; odizolovává procesy od sebe navzájem. Na druhou stranu je někdy potřeba, aby spolu mohly procesy spolupracovat.

Klasické procesy mají oddělené adresní prostory. Pokud spolu chtějí komunikovat, musí použít prostředky poskytované operačním systémem.

Multitasking může usnadnit programování – příkladem jsou například síťové servery, které obsluhují několik klientů současně. Při klasickém naprogramování pomocí jednoho procesu by tento proces byl tvořen velkou smyčkou, ve které by se přijímaly požadavky klientů a postupně se vyřizovaly. Přitom je nutné používat neblokující služby systému, protože jinak by klient připojený přes pomalou linku (nebo klient, který zhavaroval) zablokoval vyřizování požadavků od jiných klientů.



Umožňuje-li systém vytváření podřízených procesů (child process, potomek), může server fungovat tak, že při příchodu požadavku od klienta, server odštěpí (fork) podproces. Původní proces bude nadále čekat na další požadavky klientů. Potomek obslouží klienta a skončí.



Thread (lightweight process) – elementární proces, vlákno řízení = „vylehčený proces“. Některé systémy podporují tzv. multithreading – umožňují, aby se jeden „těžký“ proces skládal z více vláken řízení – threadů. Thready jednoho procesu sdílejí adresní prostor paměti a mohou spolu komunikovat pomocí sdílené paměti. Nepodporuje-li systém multithreading, znamená to, že každý proces je tvořen právě jedním threadem. Výhodou threadů je nižší režie při přepínání mezi thready a snazší spolupráce mezi thready (než mezi dvěma procesy). Každý thread má samostatný zásobník a udržuje se pro něj stav procesoru (včetně programového čítače).

6 Spolupráce mezi procesy

Dva používané mechanismy:

- sdílená paměť
- zasílání zpráv

Některé operační systémy podporují oba mechanismy.

Porovnání:

Sdílená paměť: jednodušší programování, mocnější – programátor má více prostředků, zpravidla i jednodušší implementace.

Zasílání zpráv: flexibilnější, je možné použít i pro komunikaci mezi procesy běžícími na různých procesorech nebo počítačích.

6.1 Zasílání zpráv

Zaslání zprávy může být realizováno předáním ukazatele na zprávu ve sdílené paměti, umístěním zprávy do vyrovnávací paměti nebo do fronty zpráv buď ve sdílené paměti nebo v paměti jádra systému, přičemž systém pak zajišťuje kopírování zpráv do paměťového prostoru příjemce zprávy, možná je i komunikace prostřednictvím sítě.

Implementace komunikačního spojení pro zasílání zpráv se v různých systémech mohou lišit v následujících detailech:

- způsob vytváření spojení
- počet procesů využívajících spojení: 2 nebo více
- počet spojení mezi dvěma procesy: 1 nebo více
- kapacita spojení: kolik nezpracovaných zpráv může spojení obsahovat
  - nulová kapacita: buď musí odesilatel čekat na příjemce (rendezvous), nebo se zpráva, na níž nikdo nečeká, ztratí (aby k tomu nedošlo, je nutné použít nějaký synchronizační prostředek)
  - omezená kapacita: pokud je linka zaplněná, musí odesilatel čekat
  - neomezená kapacita: odesilatel nikdy nečeká
- velikost zpráv: pevná nebo proměnná
- jednosměrnost nebo obousměrnost spojení
- přímá (zpráva se zasílá konkrétnímu procesu, je přijímána od konkrétního procesu) nebo nepřímá (zpráva se zasílá a přijímá prostřednictvím poštovní schránky) komunikace
- symetrická nebo nesymetrická komunikace
- asynchronní nebo synchronní komunikace: odesilatel musí při synchronní komunikaci čekat na odpověď (použito např. pro RPC – remote procedure call)
- automatické nebo explicitní („ruční“) použití vyrovnávacích pamětí
- zasílání kopie nebo reference

Ošetření chyb při zasílání zpráv musí zahrnovat tyto situace:

- jeden z partnerských procesů skončil
- ztráta zprávy
- duplicita zprávy
- zkomolení zprávy

6.2 Sdílená paměť

Sdílená paměť je paměť, do které má přístup více procesů. Může být použita pro komunikaci mezi procesy.

6.2.1 Souběh – race condition

Souběh (race condition) je situace, kdy při přístupu dvou nebo více procesů ke sdíleným datům dojde k chybě, přestože každý z procesů samostatně se chová korektně. K chybě dochází díky tomu, že data jsou modifikována některým procesem v době, kdy s nimi jiný proces provádí několik operací, o kterých se předpokládalo, že budou provedeny jako jeden nedělitelný celek.

Datům, která jsou sdílena několika procesy tak, že při přístupu k nim by mohlo dojít k souběhu, se říká kritická oblast.

Kritická sekce je nejmenší část programu, ve které se pracuje s daty v nějaké kritické oblasti, a která musí být provedena jako jeden celek.

Zejména u složitějších datových struktur (obousměrné spojové seznamy, složité dynamické struktury uložené v souborech apod.) dochází často k tomu, že v určitém stadiu zpracování jsou data dočasně nekonzistentní. Pokud v tom okamžiku dojde k přepnutí kontextu na proces, který tato data také používá, může nastat souběh.

Příklady souběhu:

1) Při „současně“ provedeném vkladu a výběru peněz v bance realizovaném na víceúlohovém počítači může dojít vlivem souběhu ke ztrátě vkladu:

```
1. proces (výběr)                       2. proces (vklad)
pom1:=konto;
pom1:=pom1-10000;
                -> (context switch) ->
                                        pom2:=konto;
```

```
                                        pom2:=pom2+20000;
                                        konto:=pom2;
                <- (context switch) <-
konto:=pom1;
```

2) Dva procesy se snaží vytvořit soubor. Pokud si zvolí stejné jméno souboru, může dojít k tomu, že první proces zjistí, že soubor tohoto jména neexistuje. Pak je přerušen druhým procesem, který také zjistí, že soubor neexistuje, vytvoří jej a zapíše do něj nějaké data. První proces potom provede operaci vytvoření souboru, čímž data zapsaná do souboru prvním procesem smaže:

```
1. proces                                           2. proces
if not file_exists('NAME')
then begin
                        -> (context switch) ->
                                                    if not file_exists('NAME'
                                                    then begin
                                                      assign(File, 'NAME');
                                                      rewrite(File);
                                                      write(File, something);
                                                      close(File);
                                                    end;
                        <- (context switch) <-
  assign(File, 'NAME');
  rewrite(File);
  write(File, something);
  close(File);
end;
```

Pokud jsou použity jednoduché datové struktury, lze souběhu zabránit takovým naprogramováním, že data jsou po dokončení každé strojové instrukce konzistentní:

```
1. proces (výběr)                       2. proces (vklad)
konto:=konto-10000;
                -> (context switch) ->
                                        konto:=konto+20000;
                <- (context switch) <-
```

V případě vytváření souborů může operační systém poskytovat službu, která vytvoří soubor; pokud však soubor daného jména existuje, pokus o jeho vytvoření

skončí chybou. Jestliže je provedení této služby nepřerušitelné, k souběhu nemůže dojít.

U složitějších datových struktur (například obousměrný spojový seznam) však není možné dodržet podmínku, že každá modifikace dat se provádí jedinou strojovou instrukcí nebo jediným nepřerušitelným voláním systému.

6.2.2 Problém kritické sekce

Problémem kritické sekce je umožnit přístup ke kritické oblasti procesům, které o to usilují, při dodržení následujících podmínek:

- výhradní přístup; v každém okamžiku smí být v kritické sekci nejvýše jeden proces
- vývoj; rozhodování o tom, který proces vstoupí do kritické sekce, ovlivňují pouze procesy, které o vstup do kritické sekce usilují; toto rozhodnutí pro žádný proces nemůže být odkládáno do nekonečna; nedodržení této podmínky může vést například k tomu, že je umožněna pouze striktní alternace (dva procesy se při průchodu kritickou sekcí musí pravidelně střídat)
- omezené čekání; pokud jeden proces usiluje o vstup do kritické sekce, nemohou ostatní procesy tomuto vstupu zabránit tím, že se v kritické sekci neustále střídají – mohou do této kritické sekce vstoupit pouze omezený počet krát (zpravidla pouze jednou)

Pokud o přístup do kritické sekce usiluje některý proces v době, kdy je v ní jiný proces, případně o přístup usiluje v jednom okamžiku více procesů, je nutné některé z nich pozdržet. Toto pozdržení je možné realizovat smyčkou. Toto tzv. aktivní čekání (busy waiting) však zbytečně spotřebovává čas CPU – je možné čekající proces zablokovat a obnovit jeho běh až v okamžiku, kdy proces, který je v kritické sekci, tuto sekci opustí.

6.3 Prostředky pro zajištění výlučného přístupu

6.3.1 Zákaz přerušení

Zákaz přerušení znemožní přepnutí kontextu. Nebezpečné – proces může zakázat přerušení a zhavarovat nebo se dostat do nekonečného cyklu → celý systém je

mrtvý → u většiny systémů může přerušení zakázat pouze jádro OS, což je zajištěno tak, že zákaz přerušení je privilegovaná instrukce. OS zpravidla vnitřně používá zákaz přerušení, aby zajistil nedělitelnost posloupností instrukcí používaných při implementaci jiných synchronizačních konstrukcí.

6.3.2 Instrukce Test and set lock

Instrukce typu „test and set lock“ (TSL). Instrukce nastaví proměnnou Lock (zámek) typu boolean na true (uzamčeno) a vrátí původní hodnotu této proměnné. Celá akce musí být nedělitelná (nepřerušitelná). Na počítačích, které instrukci TSL nemají, je možné ji implementovat s použitím zákazu přerušení:

```
function TestAndSet(var Lock: boolean): boolean;
begin
  DisableInterrupts;
  TestAndSet:=Lock;
  Lock:=true;
  EnableInterrupts;
end;
```

Instrukce se používá před vstupem do kritické sekce; po výstupu z kritické sekce se proměnná Lock běžným způsobem nastaví na false (odemčeno):

```
while TestAndSet(Lock) do { nothing } ;
... kritická sekce ...
Lock:=false;
```

Některé počítače, které instrukci TSL nemají, mohou místo ní použít instrukci, která prohodí obsah registru s obsahem proměnné v paměti (SWAP nebo XCHG):

```
function Swap(var a, b: boolean);
var
  Temp: boolean;
begin
  DisableInterrupts;
  Temp:=a;
  a:=b;
  b:=Temp;
  EnableInterrupts;
end;
```

Použití této instrukce vypadá takto:

```
repeat
  x:=true;
  Swap(Lock, x);
until x=false;
... kritická sekce ...
Lock:=false;
```

Instrukce TSL (případně SWAP a XCHG) lze použít pro zajištění výhradního přístupu. Samy o sobě však nesplňují podmínku omezeného čekání a neodstraňují aktivní čekání, mohou být však základem konstrukcí, které tyto nedostatky nemají.

6.3.3 Semafory

Instrukce „test and set lock“ můžeme zobecnit takovým způsobem, že dvoustavovou proměnnou typu boolean nahradíme čítačem (proměnnou typu integer). Toto zobecnění popsal E. W. Dijkstra v roce 1965. Operace nazval P a V, podle dánských slov proberen (testovat) a verhogen (zvětšit). Některé prameny tyto operace nazývají down a up. Implementace v jádře systému (s použitím zákazu přerušení) v případě použití aktivního čekání vypadá takto:

```
procedure Down(var S: semaphore);
begin
  DisableInterrupts;
  while S<=0 do begin
    EnableInterrupts;
    DisableInterrupts;
  end;
  S:=S-1;
  EnableInterrupts;
end;
```

```
procedure Up(var S:semaphore);
begin
  DisableInterrupts;
  S:=S+1;
  EnableInterrupts;
end;
```

Semafory se používají podobně jako instrukce „test nad set“ – před vstupem do kritické sekce se vyvolá Down, po výstupu Up. Semafory popsané výše také nesplňují podmínku omezeného čekání a neodstraňují aktivní čekání.

6.3.4 Synchronizační prostředky odstraňující aktivní čekání

Odstranit aktivní čekání je možné tím způsobem, že pokud proces nemůže vstoupit do kritické sekce, je mu odebrán procesor a běh procesu je dočasně blokován. Aby při uvolnění kritické sekce bylo známo, zda a který proces odblokovat, přiřadí se ke každému semaforu nebo proměnné která funguje jako zámek používaný instrukcí „test and set lock“, fronta čekajících procesů:

```
type Semaphore = record
  value: integer;
  queue: list of process;
end;
```

Předpokládejme, že jsou k dispozici operace Enqueue pro zařazení objektu do fronty a Dequeue pro vyřazení objektu z fronty. Operace Sleep zajistí, že aktuální proces bude zablokován (tj. jeho převeden do stavu blocked); operace Wakeup zajistí odblokování procesu (převedení do stavu ready).

```
procedure Wait(var S: Semaphore);
begin
  DisableInterrupts;
  S.value:=S.value-1;
  if S.value<0 do begin
    Enqueue(S.queue, CurrentTask);
    Sleep;
  end;
  EnableInterrupts;
end;

procedure Signal(var S:semaphore);
begin
  DisableInterrupts;
  S.value:=S.value+1;
  P:=Dequeue(S.queue);
  if P <> nil Wakeup(P);
  EnableInterrupts;
end;
```

7 Uváznutí – deadlock

Pokud je ve víceúlohovém systému jednomu procesu přidělena tiskárna a druhému magnetická páska a potom první z procesů požádá o přidělení pásky a druhý tiskárny, žádný z těchto procesů nemůže pokračovat v běhu. Této situaci říkáme uváznutí neboli deadlock. Řešením této situace je odebrat jednomu z procesů prostředek, který požaduje druhý proces. Většina operačních systémů však násilné odebrání prostředků nedovoluje.

Uváznutí (deadlock) – je situace, kdy dva nebo více procesů čekají na událost, ke které by mohlo dojít pouze pokud by jeden z těchto procesů pokračoval (vyřešení dopravní situace couváním).

7.1 Podmínky uváznutí

K uváznutí může dojít jenom pokud jsou splněny všechny čtyři následující podmínky:

- Výlučný přístup (mutual exclusion). Existence prostředků, které jsou přidělovány pro výhradní použití jednomu procesu, tj. nesdílitelných prostředků.
- Postupné přidělování prostředků (hold and wait). Procesy nežádají o přidělení všech prostředků najednou, ale postupně. Pokud požadovaný prostředek není volný, musí proces čekat.
- Přidělování prostředků bez preempce. Přidělené prostředky nelze procesu násilím odebrat.
- Cyklické čekání.

7.2 Řešení otázky uváznutí

Strategie, jak se operační systém může vyrovnat s uváznutím:

- ignorovat je – přestože tato strategie vypadá nepřijatelně, používá ji například OS Unix; v případě uváznutí musí zasáhnout některý z uživatelů a jeden nebo více procesů násilím ukončit; v krajním případě může být nutné znovu nastartovat celý systém

- předcházet mu – zabránit splnění aspoň jedné z podmínek nutných pro vznik uváznutí
- vyhýbat se mu – systém se vyhýbá situaci, kdy by došlo k cyklickému čekání tím způsobem, že zná maximální nároky procesů na jednotlivé prostředky a před přidělením každého prostředku zjišťuje, zda existuje způsob, jak dokončit všechny procesy i když budou vyžadovat prostředky odpovídající maximálním nárokům; OS přidělí prostředek pouze tehdy, je-li to bezpečné (existuje-li způsob, jak všechny aktivní procesy zdárně dokončit)
- detekovat uváznutí a zotavit se z něj

7.3 Předcházení uváznutí

Strategie předcházení uváznutí se snaží zabránit splnění alespoň jedné z podmínek uváznutí:

7.3.1 Výlučný přístup

Prostředky, které jsou bez omezení sdílené, nemohou způsobit uváznutí. Příkladem jsou read-only soubory.

Virtualizace prostředků – používá se například u tiskáren a snímačů děrných štítků nebo pásek. Procesy nepoužívají přímo tiskárnu, ale výstupy zapisují do diskového souboru, který operační systém vytiskne později. U vstupních zařízení systém načte požadovaná data do souborů, ze kterých je pak jednotlivé procesy čtou. Tato metoda se nazývá spooling (simultaneous peripherial output on-line).

Bohužel existují prostředky, které jsou z podstaty nesdílitelné, na které není možné tuto metodu použít.

7.3.2 Postupné přidělování prostředků

Jednorázové přidělování prostředků – každý proces si vyžádá všechny prostředky, které potřebuje při svém startu a žádné další mu nebudou později přidělovány. Pokud procesu nemohou být přiděleny všechny požadované prostředky, nejsou mu přiděleny žádné a proces musí čekat.

Alternativou je strategie, při které je možné přidělovat prostředky pouze procesu,

který žádné nemá. Díky tomu může proces několikrát za dobu svého běhu vrátit všechny prostředky a pak žádat o další.

Nevýhody:

- využití prostředků je nízké, prostředky jsou vlastně přidělovány procesům, které je ve skutečnosti zatím nepotřebují
- možnost stárnutí procesů (starvation); nároky procesu, který potřebuje několik často používaných prostředků, nemusí být nikdy uspokojeny a proces tak může čekat neomezenou dobu

7.3.3 Přidělování prostředků bez preempce

Pokud je možné snadno uschovat a následně obnovit stav prostředků, může operační systém odebrat procesu prostředky, které potřebuje pro jiné procesy.

Díky nutnosti uschovat o obnovit stav prostředku, je možné tuto strategii používat pouze u prostředků, které to umožňují jako je procesor nebo operační paměť.

7.3.4 Cyklické čekání

Algoritmus, který zamezuje cyklickému čekání: Každému typu prostředku je přiděleno číslo. Pokud má proces přiděleny nějaké prostředky, může žádat pouze o takové další prostředky, jejichž číslo je větší než největší z čísel procesem už držených prostředků. O prostředky, které mají stejné číslo musí žádat najednou.

7.3.5 Shrnutí

Některé ze strategií, které se používají pro předcházení uváznutí, je možné použít pouze pro určité prostředky (sdílitelné prostředky a prostředky, jejichž stav je možné uschovat a obnovit). Ostatní strategie mohou vést k nízkému využití prostředků a ke stárnutí procesů, což může být považováno za tak velkou nevýhodu, že v mnoha operačních systémech nejsou tyto strategie používány.

7.4 Vyhýbání se uváznutí

Strategie předcházení uváznutí jsou buď příliš omezující, nebo nepokrývají všechny prostředky používané v příslušném systému. Pokud není možné zabránit prvním třem podmínkám vzniku uváznutí a není schůdné ani podstatné omezení, které klade na systém výše popsaný algoritmus předcházení cyklickému čekání, je možné použít strategii vyhýbání se uváznutí.

Při žádosti o prostředek systém kontroluje, zda existuje alespoň jeden proces, který je možné po přidělení tohoto prostředku dokončit. Po jeho dokončení opět musí existovat alespoň jeden proces, který může být dokončen a tak dále až po ukončení všech procesů. Pokud by tato podmínka nebyla splněna, systém prostředek nepřidělí.

Různé algoritmy – nejznámější je bankéřům algoritmus:

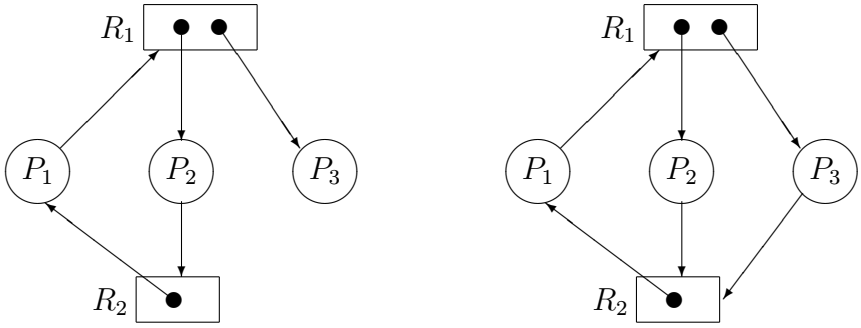
Každý proces deklaruje pro každý typ prostředku maximální množství, které bude potřebovat. Operační systém si musí při přidělování prostředků ponechat tolik prostředků, aby byl schopen uspokojit maximální požadavky alespoň jednoho procesu. Tím je zaručeno, že tento proces skončí a uvolní prostředky, které pak mohou být přiděleny dalším procesům. OS nekontroluje pouze, zda je možné dokončit jeden proces, ale všechny aktuální procesy.

7.5 Detekce uváznutí a zotavení

OS musí udržovat graf přidělení prostředků. Pokud je v grafu cyklus, došlo k uváznutí.

Situace je poněkud komplikovaná tím, že některé prostředky mohou existovat ve více exemplářích a pak procesy obvykle nežádají o konkrétní exemplář.

Na prvním obrázku je situace, kde zdánlivě existuje cyklus, ale k uváznutí nedošlo. Na druhém obrázku je uváznutí:



8 Správa paměti

8.1 Hierarchie pamětí

Akumulátor: obvykle jeden registr o velikosti rovné délce slova procesoru (8, 16, 32, 64 bitů). Jeho použití může být u některých procesorů rychlejší než použití ostatních registrů například díky kratšímu kódu instrukcí používajících akumulátor, případně některé instrukce pracují pouze s akumulátorem.

Registry: několik až několik desítek registrů o velikosti rovné délce slova procesoru.

Cache procesoru: na některých procesorech není vůbec; u Intel 8086 pouhých několik bytů instrukční fronty. Na novějších procesorech mívá rozsah stovky KB nebo několik MB. Na mikroprocesorech Intel se dělí na cache úrovně 1 (je součástí mikroprocesoru) a úrovně 2 (mimo procesor). Tzv. write-through cache – data se zapisují ihned, write-back cache data se zapisují do vnitřní paměti později.

Vnitřní paměť: ve Von Neumannově architektuře slouží pro ukládání dat i programu. U nejstarších počítačů a současných jednočipových mikrořadičů má velikost několik KB, u současných počítačů desítky až stovky MB (i gigabyty).

Diskové cache: část vnitřní paměti používaná pro data čtená v předstihu z disku (read-ahead cache) a pro opožděný zápis na disk – urychluje diskové operace.

Sekundární paměti (vnější paměti – disky): zařízení s přímým přístupem umožňující četní i zápis. Je na nich obvykle systém souborů, který umožňuje používat disky jako hierarchickou strukturu adresářů (složek, atd.) obsahujících pojmenované soubory.

Terciální paměti (zálohovací zařízení – magnetické pásky, optické disky CD,

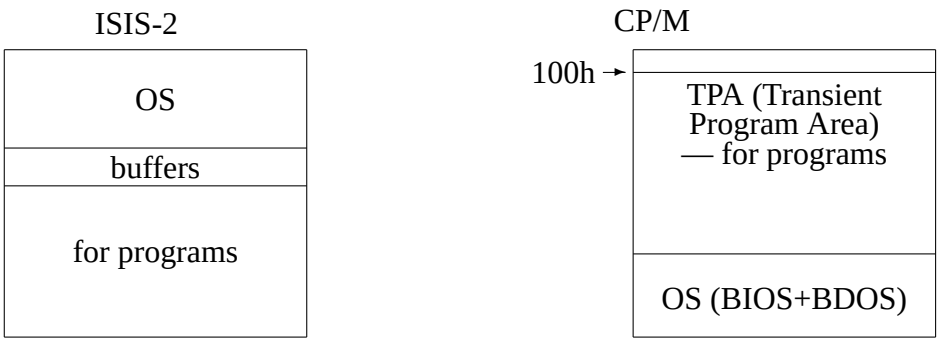
DVD): některé operační systémy automaticky přesunují soubory, které nebyly dlouhou dobu používány, z disků na zálohovací média, a v případě, že se mají použít je transparentně kopírují zpět na disky.

8.2 Metody přidělování paměti

Proces může běžet pouze v případě, že má přidělenu operační paměť. Každý operační systém proto obsahuje modul správy paměti. Tento modul zajišťuje přidělování a ochranu paměti. Další důležitou otázkou je adresování. Existují různé strategie přidělování paměti:

8.3 Přidělování veškeré volné paměti

Část paměti RAM je obsazena operačním systémem (kód, proměnné, vyrovnávací paměti), zbytek je k dispozici pro uživatelský program. V každém okamžiku je tedy v paměti nejvýše jeden uživatelský program. Tato strategie byla používána v nejstarších operačních systémech v 50. a 60. letech a znovu v 70. letech na osmibitových mikropočítačích (CP/M, ISIS-2 od Intelu).



Operační paměť ISIS-2:

- na začátku paměti OS, pak proměnné a buffery
- pak volné místo pro program

Operační paměť CP/M:

- prvních 100H byte pro OS (údaje o běžícím programu, parametry)

- paměť pro program (a volné místo)
- na konci paměti proměnné systému a buffery, BDOS a BIOS (v ROM)
- adresování: programy začínají vždy na adrese 100H → mohou být prodávány přeložené programy připravené pro spuštění
- vnitřní paměť 32, 48, 64 KB → systém musí být přizpůsoben velikosti paměti

Podle velikosti dostupné paměti, velikosti jádra OS a počtu bufferů je nutné:

- v CP/M přeadresovávat operační systém
- v ISIS-2 přeadresovávat uživatelské programy

Přeadresování provádí program zvaný locator. Locator podle tabulky adres v relativním programu změní (dle umístění) příslušné adresy na absolutní.

8.3.1 Ochrana paměti

Většina procesorů, na kterých se provozují systémy s touto strategií přidělování paměti, žádnou neumožňuje. Jinak pouze ochrana OS před přepsáním uživatelským programem (pomocí mezního registru; změna pouze v privilegovaném stavu procesoru). V uživatelském stavu není možno zapisovat na adresy větší nebo rovné obsahu mezního registru ani měnit obsah mezního registru.

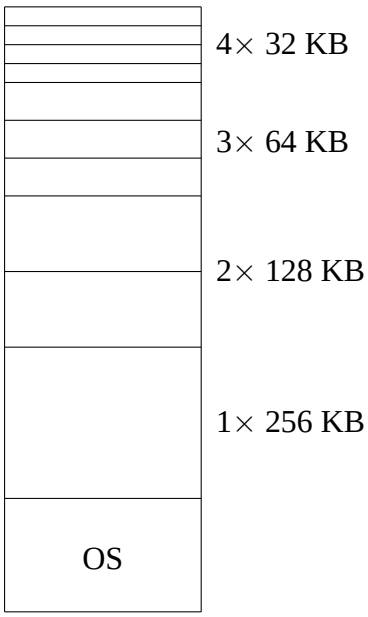
V omezené míře je možné i při této strategii přidělování paměti používat multitasking. Při přepnutí kontextu je nutné nahrát celý obsah uživatelské oblasti paměti na disk a zavést z disku obsah adresního prostoru druhého procesu.

8.4 Přidělování pevných bloků paměti

OS MFT (Multitasking with Fixed nuber of Tasks) – IBM 360 v 60. letech.

- paměť má velikost desítky KB až MB a při startu systému je pevně rozdělena na bloky (1MB se např. rozdělí na 1*256kB, 2*128kB, 2*64kB, 3*32kB, 2*16kB a 256kB pro OS)

- o každém programu jsou známy jeho paměťové nároky → OS mu přidělí blok, jehož délka je větší nebo rovna nárokům programu



Výhody:

- ve vnitřní paměti může být několik procesů současně
- jednoduchost

Nevýhody:

- špatné využití paměti (fragmentace)
- je nutné předem znát paměťové nároky,
- proces, jehož paměťové nároky jsou větší než velikost největšího bloku, má smůlu (nelze ho odstartovat – jednomu programu není možno přidělit dva sousední bloky)

8.4.1 Adresování

Není možné předem stanovit na jaké adrese bude program uložen → program musí být relokabilní. Máme dvě možnosti:

- relokační tabulka v programu = tabulka s informacemi, kde jsou v programu adresní konstanty. Při zavádění program se všechny adresní konstanty upraví podle skutečné adresy na které je program.
- program, ve kterém jsou pouze relativní adresy. U skoků to jsou autorelativní adresy (skočit o + nebo – kolik bytů). U proměnných se používá bázování – určitý registr se naplní skutečnou adresou např. začátku programu a pro odkazy na proměnné se používá místo pevné adresy hodnota bázového registru + posunutí (offset). Je velmi žádoucí, aby procesor podporoval bázování a musí mít relativní skoky.

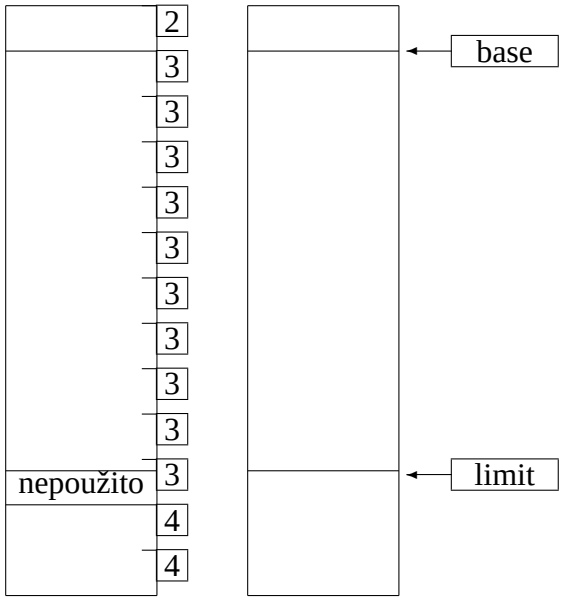
Relokace s použitím relokační tabulky:

```
for i:=1 to Number_Of_Addresses_In_Relocation_table do begin
  Address := Relocation_Table[i];
  Memory[Start+Address] := Memory[Start+Address] + Start;
end;
```

8.4.2 Ochrana paměti

Nejčastěji se používá jedna z těchto metod:

- mezní registry
- mechanismus zámků a klíčů (viz dále)



Pro použití každé z těchto metod je nutná podpora hardware → používá se ta, která je dostupná na daném procesoru.

8.4.3 Ochrana paměti pomocí mezních registrů

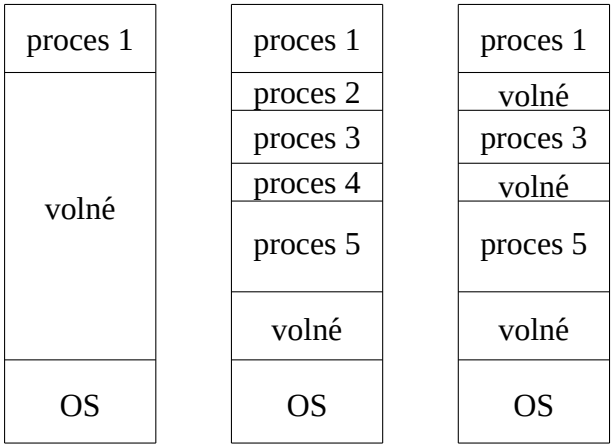
Dva mezní registry udávají nejnižší a nejvyšší dostupnou adresu. Nastavuje je OS, když předává řízení procesu. Odkaz na paměť mimo způsobí vnitřní přerušení „porušení ochrany paměti“. Nastavení mezních registrů musí být privilegovaná instrukce, jinak může program napsaný se špatným úmyslem číst nebo měnit paměťové oblasti jiných procesů.

8.4.4 Ochrana paměti pomocí mechanismu zámků a klíčů

Paměť je rozdělena na stránky pevné velikosti (např. 4 KB). Každé stránce paměti je přiřazen zámek (= celé číslo). Procesor má speciální registr, který obsahuje klíč. Proces může používat pouze ty stránky paměti, které mají zámek nastavený na stejnou hodnotu, jako je klíč. OS může používat univerzální klíč číslo 0, který umožňuje přístup k libovolné stránce paměti.

8.5 Přidělování bloků paměti proměnné velikosti

Volná paměť není pevně rozdělena, ale při startu programu se přidělí paměť podle nároků programu (resp. přidělí se celý volný blok a program vrátí, co nepotřebuje). Najdeme u MS-DOS, OS-MVT (Multitasking with Variable nuber of Tasks).



Operační paměť na počítačích PC kompatibilních v reálném režimu:

- na začátku OS, buffery, proměnné
- pak volno až do 640kB (do tohoto prostoru se zavádějí programy)
- VRAM (128kB) Video RAM
- VROM (32kB) Video ROM (u adaptérů EGA, MCGA, VGA, SuperVGA, jinak volné)
- volné místo (toto místo mohou využívat přídavné adaptéry pro své paměti RAM nebo ROM)
- ROM BIOS (64 kB)

Strategie výběru bloku:

- First fit
  - procházejí se bloky paměti a přidělí se první blok délka je větší nebo rovna požadované paměti.
- Last fit

  - vybere se poslední vyhovující

- Worst fit
  - vybere se největší volný blok; část požadované velikosti se přidělí procesu, zbytek bude volný
- Best fit
  - vybere se nejmenší vyhovující (pro přidělování bloků pevné velikosti se zpravidla používá tato strategie)

Výhody:

- jako u předchozí strategie
- lepší využití paměti

Nevýhody:

- součet paměťových nároků, které jsou současně v paměti musí být menší nebo roven velikosti paměti
- fragmentace paměti – procesy vyžadují souvislé úseky paměti. Může se stát, že není možné spustit další proces, protože má větší nároky na paměť než je velikost největšího volného bloku i přesto, že součet velikostí volných bloků je větší než paměťové nároky procesu

Pro odstranění fragmentace používají další strategie správy paměti následující metody:

- setřásání bloků; problémy s adresními konstantami lze odstranit použitím segmentace
- stránkování = programu se jeví adresní prostor jako souvislý, přestože ve skutečnosti není

8.5.1 Ochrana paměti

Pomocí mezních registrů nebo mechanismem zámků a klíčů – jako u přidělování pevných bloků paměti.

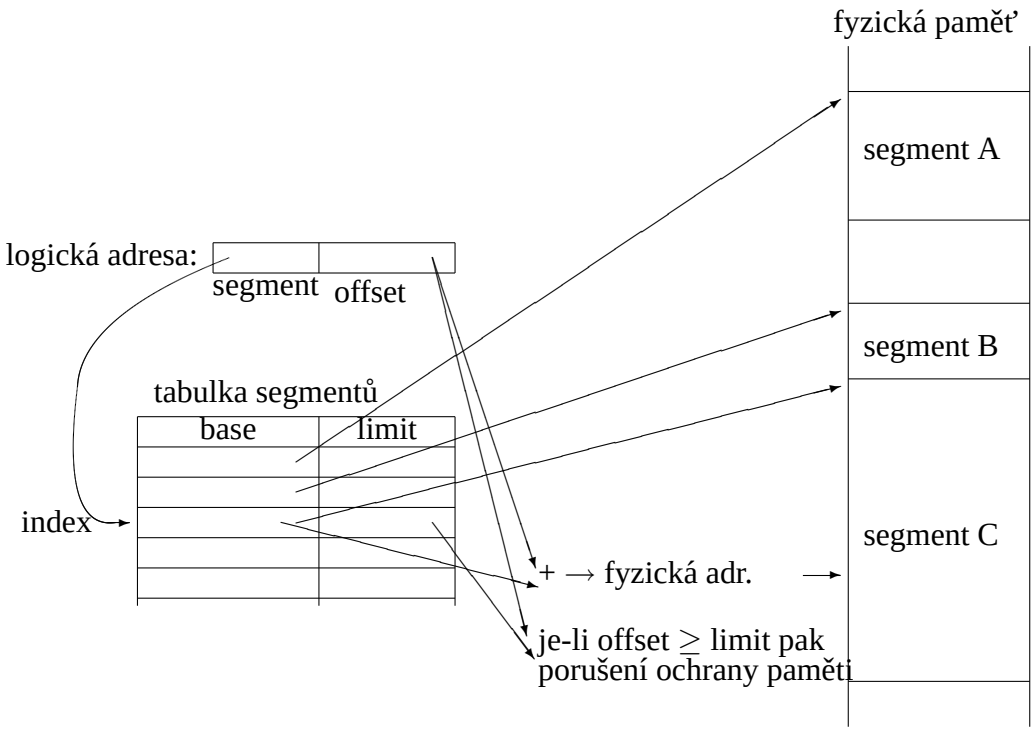
8.6 Segmentace paměti

Fyzická (skutečná) adresa v paměti se získává sečtením obsahu segment registru a logické adresy (= adresa použitá v programu). Obsah segment registru nastavuje OS a pro uživatelský program je nepřístupný. Díky tomu adresní prostor každého procesu začíná na 0 a odpadají problémy s relokací programu. Většina systémů, které používají segmentaci paměti dovoluje procesům použít více segmentů. Rozdělení na segmenty zpravidla odpovídá struktuře paměťového prostoru procesu:

- kód programu → neměnný (obsah ani délka)
- konstanty → neměnné
- inicializované statické proměnné → délka se nemění, obsah je nastaven při startu procesu (načte se z souboru, který obsahuje program), při běhu procesu se jejich obsah může měnit, délka se nemění
- neinicializované statické proměnné, délka se nemění, obsah ano
- zásobník (návratové adresy z procedur a lokální proměnné a parametry); proměnná velikost i obsah, neobsahuje díry
- halda; proměnná velikost i obsah, může obsahovat díry
- paměť pro overlaye (překryvné segmenty), dynamické knihovny

Toto rozdělení umožňuje aby procesy, které jsou řízeny stejným programem sdílely kód programu a konstanty (úspora vnitřní paměti).

U systémů se sdílenou pamětí je vhodné sdílená data uložit do zvláštního segmentu a ten zpřístupnit všem procesům, které je používají → zlepšení ochrany paměti – procesy, které používají sdílenou paměť, nemají přístup k soukromým oblastem adresního prostoru jiných procesů.



Výhody:

- je možné dynamické přemísťování segmentů za běhu procesu pro odstranění fragmentace (lze spojovat volné bloky paměti přesunutím překážejícího bloku)
- možnost dodatečně zvětšovat adresní prostor procesu
- není nutné provádět relokaci programu
- možnost sdílení segmentů

Nevýhody:

- součet nároků procesů v paměti musí být menší nebo roven velikost paměti (lze obejít odkládáním segmentů na disk) – může časově náročné
- nutná hardwarová podpora segmentace

8.6.1 Odstranění fragmentace

Procesy potřebují souvislý kus paměti → setřásání segmentů. Provádí se zpravidla v okamžiku, kdy při startu nového procesu není žádný volný blok paměti dostatečné velikosti, ale součet volných bloků stačí (některé systémy v případě, že není dostatek volného místa, odkládají adresní prostory některých procesů na disk → swapping, viz plánování procesů).

Setřásání segmentů: překopírování adresního prostoru některých procesů + změna proměnné PCB, která slouží pro naplnění segment registru. Existují různé algoritmy, které se snaží minimalizovat velikost kopírované paměti.

- některé procesory umožňují, aby každý proces mohl používat více segmentů
- segmentace umožňuje, aby běžícímu procesu byla na žádost přidělena další paměť

8.6.2 Ochrana paměti

- mezní registr na číslo segmentu
- pro každý segment mezní registr obsahující maximální povolenou adresu segmentu (limit velikosti segmentu); segmenty mají různou délku !!!

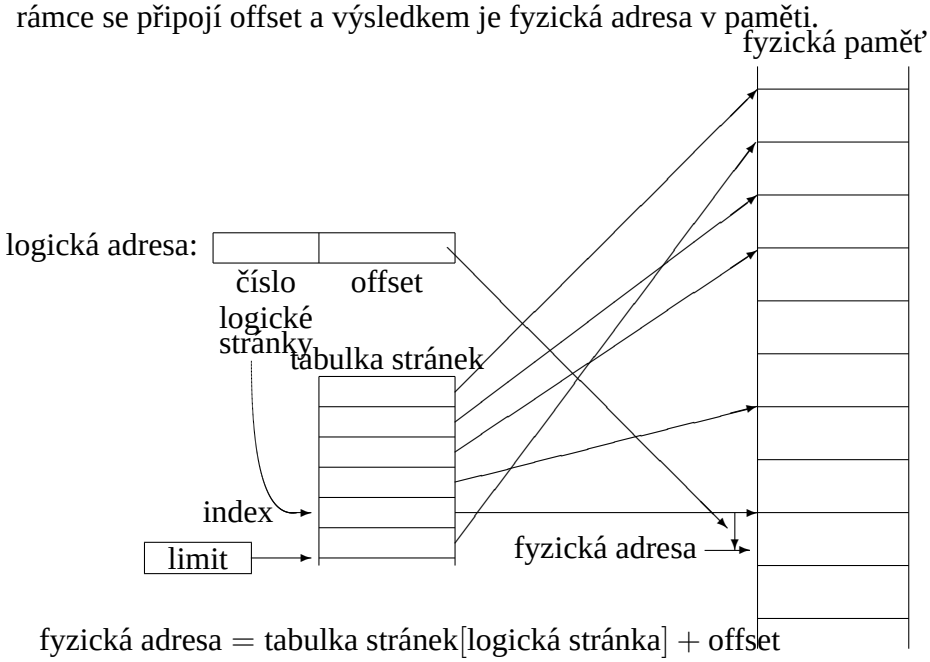
8.7 Stránkování paměti

Procesy pro svůj běh typicky požadují souvislý úsek paměti. Nutnost přidělovat souvislé úseky paměti a jejich uvolňování v libovolném pořadí podle toho, jak končí jednotlivé procesy, vede k fragmentaci paměti. Jednou z metod, jak se s fragmentací vyrovnat, je přemísťování segmentů, které však může být časově náročné. Stránkování paměti umožňuje přidělit procesu několik nesouvislých úseků paměti, a vytvořit pro proces iluzi, že tato paměť souvislá je.

Při stránkování paměti je fyzická paměť je rozdělená na rámce – frames (někdy se nerozlišuje rámec a stránka).

Logická adresa (= adresa použitá v programu) je rozdělena na dvě složky, číslo stránky a posunutí v rámci stránky (OFFSET). Velikost stránky bývá řádově kilobyty. Při velikosti stránky 4 KB je pro offset potřeba 12 bitů ($2^12 = 4K$), čili

spodních 12 bitů logické adresy je offset, zbylé bity jsou číslo stránky. Po rozkladu adresy (vše provádí procesor bez asistence programátora) na číslo stránky a offset se číslo stránky použije jako index do tabulky stránek (každý proces má svoji vlastní). V tabulce stránek je uvedeno číslo rámce ve fyzické paměti. K číslu rámce se připojí offset a výsledkem je fyzická adresa v paměti.



fyzická adresa = tabulka stránek[logická stránka] + offset
je-li číslo logické stránky $\geq$ limit pak porušení ocgrany paměti

Výhody:

- odstranění fragmentace
- není nutné přemísťování bloků paměti

Nevýhody:

- poslední stránka procesu nebývá zcela využita $\rightarrow$ velikost stránek nesmí být příliš velká (tzv. vnitřní fragmentace)
- nutné HW podpora
- součet paměťových nároků procesů v paměti nemůže překročit velikost fyzické paměti

8.7.1 Ochrana paměti

- znemožnění použít adresy mimo adresní prostor procesu → mezní registr obsahuje max. číslo stránky pro příslušný proces
- procesu není dovoleno měnit obsah tabulky stránek

8.8 Stránkování na žádost (demand paging)

Z pozorování vyplývá: většina procesů se chová tak, že po určitou dobu používá několik málo oblastí paměti a s průběhem procesu se tyto oblasti mění relativně pomalu → princip lokality paměťových odkazů. Díky tomu není nutné po celou dobu běhu procesu udržovat celý jeho adresní prostor v paměti. Adresování funguje podobným způsobem jako u obyčejného stránkování. V tabulce stránek je však pro každou stránku údaj, zda se stránka nachází v paměti nebo na disku. V případě, že je stránka na disku, je uvedeno i její umístění na disku. Pro stránkování se používá zpravidla zvláštní soubor, partition (oblast na disku) nebo dokonce disk. V minulosti se používaly bubnové paměti s mnoha hlavami, které zpracovávaly několik bitů současně (pro zvýšení rychlosti).

V případě, že se proces odkazuje na stránku, která je přítomna ve fyzické paměti, vše probíhá jako u běžného stránkování. Pokud však stránka ve fyzické paměti není (je na disku), dojde k vyvolání vnitřního přerušení „výpadek stránky“. Obslužný program přerušení musí do vnitřní paměti zavést stránku z disku, opravit odkaz v tabulce stránek, a zajistit zopakování instrukce, která výpadek stránky způsobila. Pokud je ve vnitřní paměti volné místo, použije se libovolný z volných rámců. Pokud jsou všechny rámce plné, je nutné vybrat některý z nich a přenést jej do stránkovacího souboru na disk. Existuje několik algoritmů nahrazování stránek nebo „výběru oběti“.

8.8.1 Algoritmy nahrazování stránek

Zdánlivě nejjednodušší je algoritmus FIFO. Tento algoritmus vyhodí z paměti stránku, která je v ní nejdéle. Bohužel to může být stránka, která se používá trvale, což efektivitu algoritmu snižuje. Algoritmus FIFO také vykazuje tzv. FIFO anomálií. Pokud se provede tentýž výpočet dvakrát s různě velkou vnitřní pamětí, mělo by při výpočtu s větší pamětí dojít nejvýše ke stejnému počtu výpadků stránek jako při výpočtu s menší pamětí. Při použití strategie FIFO to nemusí vždy platit. Navíc není snadné implementovat nahrazování stránek pomocí strategie FIFO. V praxi se proto používají jiné algoritmy.

Optimální algoritmus nahrazování stránek by vyhodil z paměti tu stránku, která v budoucnosti nebude použita nejdelší dobu. Předpovídat budoucnost však není možné, proto se používají algoritmy, které pro odhad budoucího chování používají chování v minulosti. Algoritmus LRU (least recently used) vyhazuje z paměti tu stránku, která nebyla nejdelší dobu použita. Implementace tohoto algoritmu může používat buď registr udávající čas posledního odkazu na danou stránku nebo frontu, na jejíž začátek se zařazuje stránka, na kterou byl právě proveden odkaz. Má-li být z paměti některá stránka vyhozena, vybere se ta, která nebyla použita nejdéle (v případě použití fronty je to poslední ve frontě). Algoritmus LRU je sice kvalitní, ale jeho hardwarová implementace je obtížná.

Proto se používá zjednodušení algoritmu LRU nazývané NUR (not used recently). V tomto případě je každému rámci přiřazen jednobitový příznak, zda byla příslušná stránka použita. Při hledání oběti se vybere ta stránka, která použita nebyla.

V algoritmech nahrazování stránek je vhodné brát v úvahu, zda byl obsah stránky změněn. V případě, že nebyl, stačí stránku pouze zahodit (její kopie je na disku). Pokud byla stránka změněna, je nutné ji nahrát na disk, což trvá přibližně stejně dlouho jako načtení nové stránky z disku. Pro tento účel bývá pro každý rámec k dispozici jednobitový příznak, který se vynuluje při zavedení stránky do vnitřní paměti a nastaví při zápisu do rámce.

Výhody:

- odstranění fragmentace
- není nutné přemísťování bloků paměti
- součet paměťových nároků procesů v paměti může překročit velikost fyzické paměti

Nevýhody:

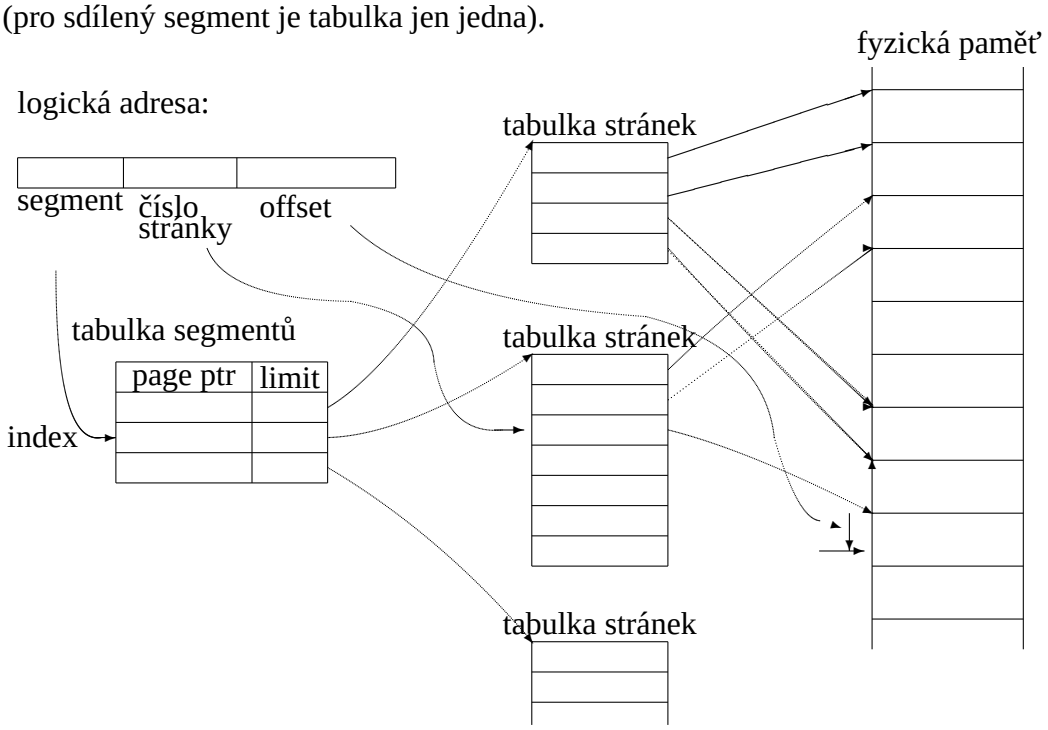
- poslední stránka procesu nebývá zcela využita → velikost stránek nesmí být příliš velká (tzv. vnitřní fragmentace)
- nutné HW podpora

8.8.2 Ochrana paměti

– jako u normálního stránkování, navíc ochrana stránkovacího souboru na disku

8.9 Segmentace se stránkováním na žádost

Logická adresa se skládá z čísla segmentu, čísla stránky a offsetu na stránce. Každý proces má vlastní tabulku segmentů. Každý segment má vlastní tabulku stránek (pro sdílený segment je tabulka jen jedna).



Číslo segmentu se použije jako index do tabulky segmentů, v tabulce segmentů se vyhledá adresa tabulky stránek, v této tabulce stránek se použije číslo stránky jako index (a zároveň se porovná, zda je menší než limit z použité položky z tabulky segmentů) a získá se fyzické číslo stránky. K němu se připojí offset a výsledkem je fyzická adresa.

Výhody:

- odstranění fragmentace
- je možné používat více paměti (virtuální paměť) než je velikost fyzické paměti
- je možné sdílet segmenty

Nevýhody:

- složitost

- nutná podpora hardware

9 Zařízení

9.1 Rozdělení zařízení

- znaková: klávesnice, displeje, terminály, tiskárny, snímače a děrovače děrné pásky a děrných štítků, myši, plottery, tablety
- bloková: disky, magnetické pásky

Některá zařízení do tohoto dělení nezapadají – paměťově mapované displeje, časovače.

Rozhraní I/O zařízení poskytované operačním systémem by mělo být jednotné pro všechna zařízení do takové míry, jak je to jenom možné.

9.2 Správa zařízení

Účelem správy je zabezpečit přístup k zařízení (pro OS) standardním způsobem. Zpravidla se požaduje transparentnost přístupu k zařízením (tj. stejný přístup jako k souborům → až při běhu programu lze určit, kam výstup půjde). Dalším úkolem je zajistit přidělování a sdílení zařízení, ochrana zařízení (přístupová práva různá pro různé uživatele).

Přidávání nových druhů zařízení:

- zásah do jádra OS (v Unixu pouze takto)
- instalovatelné ovladače zařízení (časté v MS-DOSu)
- mnoho OS kombinuje obě možnosti

Ovladače zařízení mají tři části:

- obslužný program přerušení
- část závislá na zařízení
- část nezávislá na zařízení (správce vyrovnávací paměti, pojmenovávání zařízení apod.)

9.3 Časovač

Časovač umožňuje plánovat události → proces může vyvolat službu OS, která jej zablokuje („uspí“) do uplynutí požadovaného času. Též se používá pro některé systémové akce (přidělování časových kvant, zastavování motorů disketových mechanik apod.).

Není možné předpokládat, že počítač má dostatek hardwarových (dále jen HW) časovačů pro všechny účely. Zpravidla je k dispozici jen jeden časovač a SW (programová) časová fronta. Hardwarový časovač se pak používá pro odměření času do nejbližší události ve frontě. Do fronty jsou události řazeny tím způsobem, že se zjistí po které události má k nově přidávané události dojít a spočítá se a zaznamená čas (který má uplynout mezi těmito dvěma událostmi). Při vynulování časovače se vybere první událost z fronty, provede se příslušná akce a hardwarový časovač se přeprogramuje na čas následující události.

9.4 Disky

Disky fungují jako vstupní i výstupní zařízení. Na disky se zpravidla ukládají data v podobě souborů, které jsou odlišeny jménem.

9.4.1 Rozhraní pro připojení disků

ATAPI – na současných PC nejpoužívanější.

SCSI [skazi] – dražší; poskytuje vyšší výkon a méně zatěžuje CPU. Používané zejména na serverech. Umožňuje připojení 7 zařízení.

Obě základní rozhraní mají několik variant s různým výkonem.

9.4.2 Konstrukce disku

Magnetické disky jsou tvořeny několika kruhovými deskami umístěnými na společné ose. Desky rotují; u starších disků byla rychlost otáčení 3600 ot/min, u novějších 5400, 6000, 7200, 10000, 15000 ot/min. U pevných disků jsou desky kovové, diskety jsou tvořeny jedním kotoučem z ohebné plastické hmoty. Jednotlivé desky jsou z jedné nebo obou stran pokryty magnetickou vrstvou. Pro přístup (čtení nebo zápis) ke každé vrstvě (povrchu) se používá jedna magnetická hlava, která při otáčení disku opisuje na povrchu kružnici, jejíž průměr závisí na

vystavení hlavy. Na každém povrchu se tím vytváří soustava soustředných kružnic nazývaných stopy (diskety mívají několik desítek stop, pevné disky stovky až tisíce). Hlavy pro jednotlivé povrchy jsou pevně spojeny, takže v jednom okamžiku (při určitém vystavení hlav) jsou zároveň přístupné stejné stopy na všech plotnách (tzv. cylindr, válec).

Stopy jsou rozděleny na sektory. V současné době se používají zpravidla sektory pevné délky o velikosti 128 až 4096 bytů (u počítačů PC 512 bytů). Stopy blíže středu disku jsou kratší, přesto se zpravidla na všechny stopy zaznamenává stejný počet sektorů (záznam u středu disku je hustší). Proto se u některých disků se na stopách blíže středu snižuje záznamový proud (tzv. prekompenzace záznamu). Sektor je nejmenší část disku, kterou je možné číst nebo zapisovat. Pokud má být změněn jediný byte, musí OS zajistit načtení celého sektoru, změnu příslušného bytu a zpětný zápis sektoru na původní místo.

Při přístupu k určitému sektoru musí disková mechanika vystavit hlavy na požadovaný válec stopu a počkat až požadovaný sektor projde pod hlavou.

9.4.3 Výměnné disky

Diskety. V 70. letech osmipalcové s kapacitou 128 nebo 256 kB. Na PC 5,25 palcové a později 3,5 palcové s kapacitou od 160 kB do 1,44 MB (kapacita 2,88 MB se neprosadila).

Bernoulliho disky, magnetooptické disky a další výměnné disky s vyšší kapacitou: ZIP (25, 100, 250, 750 MB), JAZ (1 a 2 GB), LS-120 (120 MB) – umožňuje pracovat i s disketami, atd. Často k dispozici jako externí mechaniky (na SCSI, paralelní nebo USB port).

V poslední době jsou nahrazovány elektronickými disky velikosti klíčenky s kapacitou 25 MB až 1 GB připojovanými na USB port (někdy kombinované s přehrávačem MP3 nebo použitelné jako elektronický diktafon). Stejně se připojují a chovají i digitální fotoaparáty.

9.4.4 Optické disky

CD, DVD. Plastový kotouč uvnitř s odraznou plochou, do které jsou vylisovány (CD-ROM, DVD-ROM) nebo laserem vypáleny (CD-R, DVD-R) otvory (na RW médiích je změny odraznosti dosaženo vratnou změnou krystalické struktury na amorfní). Na rozdíl od magnetických disků mají pouze jednu spirálovou stopu od

prostředka ke kraji.

Kapacita CD: 650−870 MB (objevují se i CD mechaniky pracující s ještě většími kapacitami 1,3 GB až 2 GB; na druhou stranu CD pro singly mají průměr 8 cm a kapacitu 200 MB).

DVD mají kapacity nejčastěji 4,7 GB (jednostranné jednovrstvé), existují i formáty 8,5 GB (jednostranné dvouvrstvá), 9,4 GB oboustranné jednovrstvé a 17 GB oboustranné dvouvrstvé.

Rychlost se u CD udává v násobcích rychlosti používané pro hudební CD (150 kB/s), u DVD je základní rychlost 1353 kB/s.

9.4.5 Způsob záznamu na disky

FM, MFM, RLL.

9.4.6 Prokládání sektorů (interleave)

Přenos dat mezi řadičem disku a operační pamětí trvá určitou nenulovou dobu. Pokud se má zpracovat několik následujících sektorů, může dojít k tomu, že během zpracování prvního z nich se disk pootočí a další sektor mezitím „ujede“.

Počítač pak musí čekat celou otáčku, aby mohl tento sektor přečíst nebo zapsat. Totéž se může opakovat u dalších sektorů. V krajním případě může být pro načtení jedné stopy potřeba tolik otáček disku, kolik sektorů je na stopě.

Tento problém je možné odstranit pomocí prokládání sektorů (interleave). Sektory nejsou uloženy na stopě za sebou, ale na přeskáčku. Například je-li interleave faktor 1:3, bude pořadí sektorů na stopě následující:

```
1 7 13 2 8 14 3 9 15 4 10 16 5 11 17 6 12
```

Poměr 1:3 značí, že po sobě následující sektory (například číslo jedna a dva) jsou „ob tři“ sektory.

Interleave faktor 1:1 znamená, že prokládání není použito (pořadí 1, 2, 3, 4, 5, . . .). Velký interleave faktor se používal zejména u pomalých počítačů. V dnešní době mají disky cache paměť, která umožňuje načíst celý cylindr, takže se prokládání nepoužívá.

9.4.7 Obsluha požadavků na přístup k disku

Při vyřizování požadavku OS na disk je potřeba nejprve vystavit hlavy na příslušnou stopu a pak počkat, až bude požadovaný sektor pod hlavami. U víceúlohových systémů mohou přicházet požadavky na disk rychleji, než je možné je vyřizovat.

Vyřizování požadavků v pořadí, jak přicházejí (tzv. FIFO nebo FCFS – First In First Out, First Come First Serve), není optimální.

Používají se jiné algoritmy (oba lze modifikovat v případě, že existuje rychlý způsob, jak se dostat na stopu 0):

SSF (Shortest Seek First)

- máme-li hlavu na 50 cylindru a přijdou požadavky na 52, 80, 7, 49, 45 → SSF → bude vyřízeno v pořadí 50, 49, 52, 45, 80, 7
- dochází k diskriminaci okrajových cylindrů

Elevátorový algoritmus

- totéž co automatické výtahy (nahoru → dolů)
- pohyb hlavy je změněn na 50, 52, 80, 49, 45, 7

9.5 Diskové oblasti

Počítače PC umožňují rozdělit pevný disk na několik oblastí (partition). Každá oblast může být naformátována pro jiný typ systému souborů a použita pro jiný operační systém.

V úplně prvním sektoru na disku (tento sektor se nazývá mastar boot record – MBR) je tabulka oblastí obsahující maximálně 4 položky, z nichž každá popisuje jednu oblast (tzv. primary partition). Aby bylo možné rozdělit disk na více částí, může jedna z položek obsahovat odkaz na rozšířenou oblast (extended partition), která v prvním sektoru obsahuje další tabulku oblastí. Tabulka obsahuje odkaz na jednu další oblast (logical disk) a na část disku, jejíž první sektor obsahuje další tabulku, atd..

9.6 RAID

Umožňuje vytvořit z více disků sestavu s větší kapacitou, vyšší spolehlivostí nebo větším výkonem. Zkratka z Redundant Array of Inexpensive Disks.

RAID úrovně 0 (disk stripping) diskové oblasti mohou zabírat více než jeden disk – poněkud zvyšuje výkonnost, ale neposkytuje žádnou redundanci a díky tomu ani vyšší bezpečnost před poruchou.

RAID úrovně 1 (disk mirroring – zrcadlení disků) všechna data jsou ukládána na dva disky; pokud jeden z nich selže, OS používá druhý. Vystačí již se dvěma disky. Nevýhodou je 100% redundance.

RAID 0 + 1 – kombinuje mirroring a stripping.

RAID úrovně 2 (bit stripping a používání samoopravných kódů – error correcting code = ECC) každý byte dat je rozdělen na několik disků. Navíc je jeden disk obsahující bity pro samoopravný kód. Při poruše jednoho disku, řadič disků nebo OS dopočítá původní data. Je komplikovanější než RAID úrovně 1, ale má nižší redundanci; v praxi se však místo něj používá RAID úrovně 3.

RAID úrovně 3 – využívá kontrolní součty sektorů, které jsou na každém normálním disku. Pokud je chyba pouze na jednom disku lze data dopočítat. Stačí 1 disk pro kontrolní součty na 4 až 8 datových disků.

RAID úrovně 4 a RAID úrovně 5 – podobný RAID úrovně 3, ale používá blok stripping s paritními bloky; RAID úrovně 4 používá jeden vyhrazený disk pro kontrolní součty (ten je nejvíce zatěžován, protože se používá při každém zápisu i čtení), RAID úrovně 5 má rozloženy paritní a datové bloky na různých discích.

RAID úrovně 6 – jako RAID úrovně 5, ale používá více disků s paritními bloky. Může pracovat i při poruše více než jednoho disku.

Pro plné využití výhod RAID jsou některé servery vybavovány tzv. hot swap disky – disku lze vyměnit za chodu počítače bez nebezpečí zničení disků nebo počítače – při poruše disku je možné vadný disk nahradit dobrým za provozu a systém zajistí regeneraci dat, aby byl připraven na poruchu dalšího disku.

10 Správa souborů

10.1 Struktura souborů na disku

Moderní OS používají téměř výhradně hierarchický systém souborů (adresáře, podadresáře, . . .). Ve starých OS nebyl (CP/M, OS-MFT, OS-MVT, IBM v 60. letech, MS-Windows).

Existují různé přístupy k více diskům:

- oddělené systémy souborů na jednotlivých discích (A:, B:, C: – MS-DOS, podobně MacIntosh, VMS, IBM OS-MVT, CP/M)
- UNIX → mountování filesystémů. Na jednom disku je kořenový adresář. V hierarchické soustavě adresářů se na některý adresář pověsí celý disk (přepneme-li se do tohoto adresáře, ocitneme se na jiném fyzickém disku).

Ve starších OS byla značná omezení na jména souborů: krátká jména (CP/M maximálně 6+3 znaky, MS-DOS 8+3, Unix 14), nerozlišují se malá a velká písmena, množina znaků, které mohou tvořit jméno souboru je značně omezená. Novější OS často rozlišují malá a velká písmena, povolují speciální znaky (mezera, +, !, :) a národní znaky (ěščřžýáié. . .). V pojmenovávání souborů vývoj směřuje k dlouhým jménům (desítky znaků).

10.2 Systém souborů FAT

FAT (File Allocation Table)

10.3 Unixové systémy souborů

Souborové systémy používané unixovými systémy prošly dlouhým vývojem. U části z nich lze vysledovat společné vlastnosti, které vycházejí ze systému souborů v Unixu System V.

Rozdělení svazku:

- Boot area – oblast pro zavádění OS

- Superblock – obsahuje popis systému souborů
- Inode blocks – obsahuje informace o souborech (kromě jmen)
- Data blocks – data obsažená v souborech a adresářích

Superblok:

- Velikost systému souborů v blocích
- Počet bloků v seznamu inodů
- Počet volných bloků
- Počet volných inodů
- Seznam volných bloků
- Seznam volných inodů

Adresáře tvoří v unixových systémech od počátku hiearchickou (stromovou) strukturu. Na rozdíl od systému souborů FAT obsahují adresáře v unixu pouze seznam dvojic (jméno, číslo inode) a všechny informace o souboru (kromě jména) jsou umístěny v inode.

Vývoj struktury adresáře:

- položky pevné délky 16 bytů: 2 byty číslo inode, max 14 bytů jméno souboru (podadresáře, apod.)
- položky proměnné délky: jméno souboru za končené znakem s kódem 0 číslo inode – neefektivní pro adresáře obsahující velké množství položek desetitisíce a více
- adresář organizovaný jako vyhledávací strom

Struktura inodu:

- Mode – druh souboru a práva
- Owner – vlastník

- Group – skupina
- Time stamps – časové informace
  - atime – čas posledního přístupu
  - mtime – čas poslední změny obsahu souboru
  - ctime – čas poslední změny informací o souboru
- Size – velikost souboru
- Reference count – počet odkazů na soubor
- Direct Blocks – přímé odkazy na datové bloky souboru
- Single Indirect – odkazy na blok s odkazy na datové bloky souboru
- Double Indirect
- Tripple Indirect

10.4 Přístupová práva k souborům

Přístupová práva k souborům jsou důležitým prvkem bezpečnosti víceuživatelských systémů.

10.4.1 Přístupová práva v OS Unix

Operační systém Unix používá poměrně jednoduchý systém souborových práv. Každý objekt v systému souborů má přiřazen jednoho vlastníka a jednu skupinu uživatelů. Tyto dva údaje umožňují přiřadit různá práva k danému objektu pouze třem různým kategoriím uživatelů:

- vlastníkovi
- uživatelům patřícím do zvolené skupiny
- ostatním uživatelům

Každá z uvedených kategorií může mít přidělena libovolnou kombinaci práv `r`, `w` a `x` – např.:

Vlastník	Skupina	Ostatní
`rwx`	`r-x`	`---`

Vlastník nepřebírá práva skupiny a ostatních. Význam těchto práv se poněkud liší u souborů a u adresářů.

V některých Unixech může být uživatel členem více skupin současně, v jiných pouze může mezi skupinami přecházet příkazem `newgrp`.

Práva k souborům:

- `r` čtení obsahu souboru
- `w` změna obsahu souboru
- `x` spouštění souboru (všechny programy musí mít `x`)
  pro mazání a přejmenovávání souboru stačí mít práva `wx` v adresáři

Soubor, který by měl práva uvedené v příkladu výše, může vlastník číst, zapisovat (měnit) i spouštět (jedná se o program), skupina může číst a spoustět, ostatní nemohou nic.

Práva k adresářům:

- `r` lze pouze přečíst jména souborů a podadresářů daného adresáře (nic jiného, nelze ani zjistit, zda se jedná o soubory nebo podadresáře)
- `w` samo o sobě není k ničemu; spolu s právem `x` umožňuje vytvoření, přejmenování a zrušení souborů a prázdných podadresářů v adresáři
- `x` umožňuje přepnout do adresáře, čtení a zápis do souborů, měnit vlastníka a práva souborů a podadresářů, není-li právo `r`, je nutné znát jména souborů v adresáři

10.4.2 Rozšířená práva v Unixu

Práva setuid a setgid.

Název pochází ze slov nastavit uid a gid (User Identifier = číslo uživatele, Group Identifier = číslo skupiny).

Jestliže má program nastaveno právo setuid (místo práva x pro vlastníka se vypisuje s), má uživatel při jeho spuštění stejná práva jako vlastník programu. Podobně funguje setgid pro skupinu.

Příklad použití: v Unixu jsou informace o uživatelích (včetně zakódovaných hesel) uloženy v souboru `/etc/passwd`. Jeho vlastníkem je root (správce systému) a práva jsou `rw-r--r--`. Soubor tedy může kdokoli číst, měnit jej však smí pouze root. Aby si uživatelé mohli sami měnit hesla, má program, který se používá pro změnu hesel, nastaveno právo setuid (a jeho vlastníkem je root).

Výše zmíněný program musí být ale spolehlivý a dělat jen to co má, jinak narušíme bezpečnost systému (např. jakýkoliv uživatel bude moci změnit jiným uživatelům hesla a pak zneužít jejich konta).

U sdílených adresářů pro dočasné soubory se používá sticky bit. Adresáře mají všechna práva pro všechny, což umožňuje každému vytvářet soubory, k těmto souborům přistupovat podle jejich práv, ale i mazat libovolné soubory. Pokud má adresář nastavený sticky bit, může každý uživatel mazat pouze soubory, jichž je vlastníkem.

U některých Unixů se používá tzv. ACL (Acces Control List). Pro každý soubor a adresář existuje seznam uživatelů a skupin, u každého jsou uvedena práva (podobně jako u Novellu).

10.4.3 Přístupová práva v OS Novell Netware 3.x

Právo	Adresář	Soubor
S (supervisory)	všechna práva pro adresář a celý podstrom neomezen maskou zděděných práv, může přidělovat práva ostatním	všechna práva k souboru
R (read)	otevřít a číst soubory (není-li F, musí znát jméno souboru)	otevřít a číst soubor
W (write)	otevřít a zapisovat do exist. souborů	zapisovat do souboru
C (create)	vytvářet soubory a adresáře, zapisovat do nově vytvořených souborů	obnovit soubor (salvage) po smazání
E (erase)	smazat soubory nebo prázdné adresáře	smazat soubor
M (modify)	měnit jména a atributy souborů a podadresářů (ne mazat a měnit jejich obsah)	→ totéž
F (file scan)	je vidět obsah adresáře po DIR	vidět soubor při DIR
A (acces control)	umožňuje měnit práva ostatních uživatelů (s výjimkou S); i ta co sami nemáme	→ totéž

U každého adresáře a souboru lze definovat seznam tvořený dvojicemi: uživatel nebo skupina – práva. Uživatel může být členem více skupin. Výsledná práva

jsou sjednocením práv uživatele a všech skupin, kterých je členem. Navíc se práva přebírají z nadřízeného adresáře (dědění). Toto přebírání práv je možné ve vybraných adresářích omezit tzv. maskou dědičných práv.

10.5 Práva ve Windows NT a Windows XP

Lze používat pouze na diskových oddílech se systémem souborů NTFS.

Full Control	dtto
Traverse Folder	Execute File
List Folder	Read Data
Read Attributes	dtto
Read Extended Attributes	dtto
Create Files	Write Data
Create Folders	Append Data
Write Attributes	dtto
Write Extended Attributes	dtto
Delete Subfolders and Files	dtto
Delete	dtto
Read Permissions	
Change Permissions	
Take Ownership	

10.6 Atributy souborů

Obvykle bitové příznaky popisující, jaké operace lze se souborem provádět. V MS-DOSu existují pouze čtyři příznaky:

R	read-only	jen pro čtení
H	hidden	skrytý
S	system	systémový
A	archive	nebyl archivován

Soubory s atributem read-only nelze měnit ani mazat. Atribut však lze bez problémů odstranit pomocí služby OS nebo příkazem `ATTRIB`.

Atribut hidden a system se obvykle nastavují současně. Soubory s těmito atributy se považují za nepřemístitelné, tzn. že je programy například pro defragmentaci disku nikam nekopírují.

Atribut archive se automaticky nastavuje při jakékoli změně souboru. Archivační

programy umí na žádost archivovat pouze soubory, které mají tento atribut nastavený (při tzv. diferenciálním nebo inkrementálním zálohování).

10.7 Atributy souborů v systému Novell Netware

Atributy pro soubory i adresáře:

H	hidden	skrytý soubor, nevypíše se při DIR
S	system	systémový soubor, nevypíše se při DIR
D	delete inhibit	nelze smazat
R	rename inhibit	nelze přejmenovat
P	purge	nelze obnovit po smazání

Atributy pouze pro soubory:

A	archive	nebyl archivován
Ro	read only	pouze pro čtení
Rw	read write	pro čtení i pro zápis
X	execute only	nelze kopírovat, pouze spustit
I	indexed	pro urychlení přístupu k velkým souborům
T	transactional	lze definovat transakce = sady akcí, které se buď provedou celé, nebo se soubor automaticky obnoví do původního stavu
S	shareable	sdílitelný soubor → může ho otevřít více uživatelů současně

Na rozdíl od MS-DOSu může být pro uživatele příznak read-only nepřekonatelný (pokud uživatel nemá oprávnění M – modify).

10.7.1 Atributy souborů v Linuxu

Atributy lze používat na svazcích se souborovým systémem typu ext2:

A	don’t update atime (neaktualizovat čas posledního přístupu)
S	synchronous updates (provádět synchronní zápis)
a	append only (soubor lze pouze připisovat na konec)
c	compressed (komprimovaný)
d	no dump
i	immutable (soubor nelze měnit)
j	data journalling (soubor slouží pro žurnálování)
s	secure deletion (před smazáním bude obsah soubor několikrát přepsán – aby nebylo možné data obnovit)
u	undeletable (soubor nelze smazat)

10.8 Další informace o souborech

OS zpravidla o souboru udržuje další informace:

10.8.1 Datumy a časy

OS MS-DOS udržuje pouze informaci o datumu a času poslední změny souboru, jiné OS více časů:

- vytvoření
- poslední změny
- posledního přístupu

10.8.2 Typy souborů

Jsou používány na počítačích Macintosh OS Finder (MacOS).

- 4 znaky typ souboru + 4 znaky program, který jej vytvořil
- tyto znaky nejsou součástí jména souboru
- určuje, jakou aplikací se má soubor zpracovávat

10.8.3 Soubory sestávající z několika proudů

V některých systémech jsou soubory členěny na datovou část a resource část (obsahující texty, obrázky, ikony, dialogy). Tyto OS zpravidla umožňují změnu resource části programu bez nutnosti nového překladu programu. U počítačů Macintosh skutečně dvě části souboru; programy v MS-Windows je vše v jednom souboru.

Souborový systém NTFS používaný u Windows NT, Windows XP a Windows 200x umožňuje, aby se každý soubor skládal z několika proudů

10.8.4 Odkazy v systému souborů

V Unixu odkaz (link) – existují symbolické (na jméno) a pevné (na konkrétní soubor).

Na počítačích Macintosh a ve Windows zástupci.

- dva nebo více odkazů na tentýž soubor nebo adresář
- odkazy mohou být stejného nebo i různého jména

10.8.5 Generační soubory

Udržují se i starší verze souborů. Pro odkaz na soubor, lze používat buď pouze jméno (odpovídá nejnovější verzi souboru) nebo jméno;generace. V OS VMS jsou generační soubory součástí systému. V Unixu se řeší programově (RCS – Revision Control System).

10.9 Sdílení souborů

V MS-DOSu se zavedeným programem `SHARE.EXE` a ve Windows lze při otvírání souboru stanovit:

- způsob přístupu z tohoto programu (procesu) – OPEN FOR:
  - read – pouze čtení
  - write – pouze zápis

  - read & write – čtení i zápis

- jaké operace s tímto souborem znemožnit ostatním procesů po dobu jeho otevření – DENY:
  - none → ostatní mohou číst i zapisovat
  - read → znemožnit ostatním čtení
  - write → znemožnit ostatním zápis
  - all → znemožnit čtení i zápis

10.10 Uzamykání souborů a jejich částí

Je podporováno mnoha OS (MS-DOS a Windows se `SHARE.EXE`, Unixy).

Při práci se souborem lze uzamknout celý soubor nebo jeho určitou část. Ostatní procesy nemohou po dobu uzamčení obsah příslušné oblasti měnit (nebo ani číst). Často se používá například pro přístup k databázím (příklad: databáze místenek přístupná z několika předprodejů v různých městech).

10.11 Diskové kvóty

Omezení prostoru na disku (discích) zabraného jedním uživatelem. Používá se u NOVELL, Unix, VMS. Často se používá tvrdá a měkká kvóta. Tvrdou nelze v žádném případě překročit, měkkou lze překročit na omezenou dobu.

11 Správa uživatelů

Umožňuje omezit i jiná práva uživatelů než práva k objektům v systému souborů a procesům. Velmi dobře definované např. v Novell Netware.

- Kdy se uživatel může přihlásit.
- Z jakých počítačů se může přihlásit.
- Z kolika počítačů se může přihlásit současně.
- Jaká je doba platnosti uživatelské účtu.

- Zda si uživatel může měnit heslo.
- Po jaké době si musí změnit heslo.
- Jaká je minimální délka hesla, případně další požadavky na heslo.
- Souborová práva.
- Diskové kvóty.
- Práva k účtům ostatních uživatelů.
- Práva k procesům ostatních uživatelů.

12 Informační bezpečnost

Informační technologie některých organizací mohou představovat značnou hodnotu. Samotná data mohou být tím nejcenějším co firma má. I pro jednotlivce mohou být datové soubory mimořádnou důležitost (text diplomové práce apod.).

Proto je důležité chránit data

- před ztrátou (která může být způsobena např. selháním hardware, záměrným nebo neúmyslným smazáním, apod.)
- před únikem a zneužitím (u některých dat vynuceno i zákonem o ochraně osobních údajů)
- před nežádoucí modifikací

Zároveň s ochranou dat lze chránit i programové vybavení.

13 Oblasti informační bezpečnosti

Informační bezpečnost nelze redukovat pouze na technické a programové zabezpečení – lze ji rozdělit na několik oblastí:

- Fyzické zabezpečení – vhodné umístění výpočetní techniky v prostředí bez škodlivých vlivů, co nejlépe chráněném proti požáru, vytopení, znečištění

(např. i cigaretovým kouřem), přístupu nepovolaných osob, krádeži; používání zabezpečovacích prvků, zámků, alarmů, ochranky; skladování záložních kopií na jiném místě

- Technické a programové zabezpečení – zejména zabezpečení poskytované operačním systémem a dalšími softwarovými nástroji
- Organizační (administrativní) zabezpečení – vypracovaný systém záloh, odpovědnosti za provoz IT, vypracování krizových scénářů pro případ ztráty dat nebo bezpečnostního incidentu, rozhodnutí, které systémy budou zapojeny do sítě a s jakými bezpečnostními technickými prostředky
- Personální opatření – pečlivý výběr zaměstnanců na místa správců, formulace pracovních smluv, apod.

Zajišťování informační bezpečnosti je v současnosti nepřetržitý proces – kvůli velkému rozšíření sítí, prudkému nárůstu složitosti softwaru i díky přepravě cenných a snadno zneužitelných informací (platební příkazy, hesla k bankovním účtům, utajované informace, apod.).

13.1 Přehled bezpečnostních incidentů a metod průniku

Počítačové viry – kus programového kódu (obvykle strojového kódu nebo skriptu) připojený k programu, uložený v boot sektoru disku, nebo obsažený v dokumentu, který je schopen se kopírovat do jiných programů, na jiné disky nebo do jiných dokumentů. Virus může obsahovat i škodlivý kód.

(Počítačový) červ – škodlivý kód, který se šíří elektronickou poštou nebo jiným komunikačním systémem

Trojský kůň – program obsahující škodlivý kód

Dialer – program, který přesměrovává bez vědomí uživatele modemové připojení k Internetu na čísla s vysokou tarifikací (60 Kč/min)

Back door, trap door, zadní vrátka – program, který umožňuje pozdější neautorizovaný přístup do systému

Buffer overflow (přeplnění vyrovnávací paměti) – programátorská chyba, která může způsobit, že zasláním většího bloku dat, než je očekávaná délka, budou přepsána data v paměti, což může vést až k předání řízení do zaslaného kódu

Exploit (průnik nebo program pro průnik) – využívá bezpečnostní díry k získání přístupu do systému nebo k zíkání vyššího oprávnění

Denial of service (DOS) – znemožnění fungování služby např. zahlcením serveru nebo rozšířením chybných řídicích informací (např. o směrování, přidělování adres, překladu jmen apod.)

Útok hrubou silou – útok spočívající v postupném zkoušení všech možných hesel nebo klíčů – proto je nutné používat hesla a klíče dostatečné délky a neuhodnutelná

Firewall – zařízení chránící vnitřní síť organizace před útokem z Internetu; pokud možno vyhrazený počítač s nainstalovaným programovým vybavením pro firewall, 2 nebo 3 síťová rozhraní (3 pokud má firma Internetové servery)

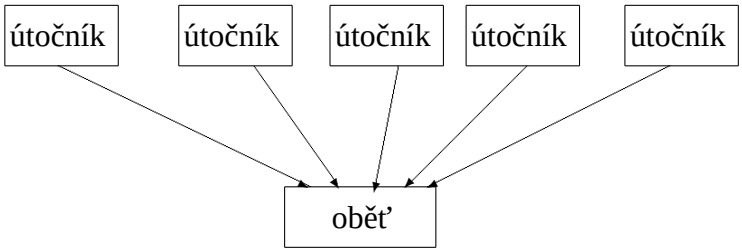
13.2 Rozdělení útoků

Útoky lze rozdělit do dvou kategorií:

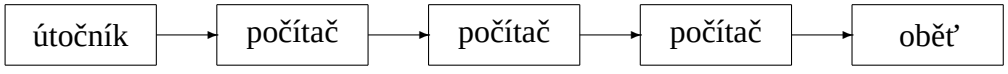
- plošné – mají za cíl buď prosté rozšíření např. viru, nebo vytvoření co nejširší základny ovládaných počítačů, které mohou být použity pro cílený útok, získání velkého množství hesel nebo jiných údajů; tyto útoky mohou být vedeny i automaticky
- cílené – s cílem dosáhnout konkrétního cíle

Využití většího množství počítačů k útoku:

- Provedení distribuovaného útoku



- Přístup přes řetěz počítačů s cílem zabránit lokalizaci a odhalení pachatele



počítače, které útočník ovládá

- opatření proti odposlechu
- ochrana proti virům
- ochrana proti červům
- ochrana proti útokům z Internetu
- ochrana proti útokům zevnitř
- ochrana proti obtěžování ostatních
- ochrana proti spamu

13.3 Bezpečnostní audit

Vychází z odhadů pravděpodobnosti různých druhů bezpečnostních incidentů, odhadů finančních ztrát, které způsobí, a jejich porovnání s cenou nevrhovaného řešení zabezpečení a odhadem jeho účinnosti.

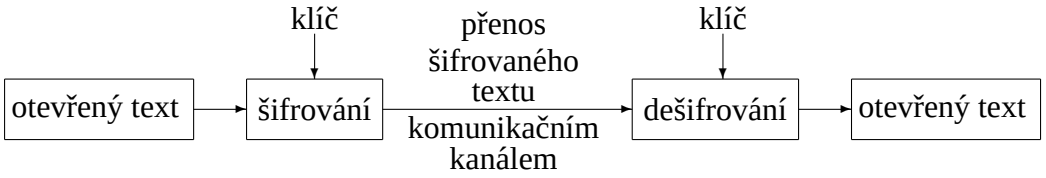
14 Kryptografie

Ve víceuživatelských systémech lze ochranu jednoho uživatele před ostatními zajistit pomocí mechanismu oprávnění. Správce systému však má obvykle neomezený přístup ke všem souborům a dalším objektům v systému. Při používání sítí lze ochranu dat pomocí mechanismu oprávnění obejít. Proto je nutné používat šifrování.

Metody šifrování lze rozdělit do tří skupin:

- Šifrování založené na utajovaném algoritmu
- Šifrování s tajným klíčem (tzv. symetrické šifrování)
- Šifrování s veřejným klíčem (tzv. asymetrické šifrování)

Šifrování založené na utajovaní šifrovacího algoritmu není pro používání na počítačích vhodné. Dříve nebo později totiž dojde k odhalení algoritmu, což umožní dešifrovat všechny zprávy, které jím byly zašifrovány. Přitom šifrované zprávy lze často luštit i bez znalosti, jaký způsob šifrování byl použit (obor, který se tím zabývá, se nazývá kryptoanalýza).



Při šifrování s tajným klíčem je klíč1 = klíč2. Bezpečnost šifry je závislá na kvalitě algoritmu a délce klíče. Pokud je klíč stejně dlouhý jako zpráva, nebude nikdy použit znovu a nedojde k jeho prozrazení, je šifra neprolomitelná. Problémem však je, jak zajistit, aby odesilatel i příjemce zprávy klíč znali, a aby se klíč nedostal do ruky nikomu jinému. Nezbytnost sdílení klíče odesilatelem a příjemcem vede k používání klíčů omezené délky, což usnadňuje luštění zpráv. Klíče s délkou 56 bitů používané standardem DES jsou v současnosti rozluštitelné. Za bezpečné se pro následujících deset let považují klíče s délkou okolo 100 bitů.

Při šifrování s veřejným klíčem využívá každý příjemce dvojici klíčů. Jeden z nich je veřejný a používá se pro šifrování, druhý (utajovaný nazývaný „soukromý“ slouží pro dešifrování. Tedy na obrázku klíč1 $\neq$ klíč2. Celý vtip je v tom, že je sice možné vytvářet libovolné množství dvojic klíčů veřejný-soukromý, ale znalost libovolného klíče z dvojice neumožňuje vypočítat klíč druhý (resp. výpočet je nerealizovatelně složitý). Výhodou šifrování s veřejným klíčem je snadná distribuce veřejných klíčů, problémem je naopak zajistit, aby nedošlo k podvržení klíče. Stupeň bezpečnosti opět závisí na délce klíče. Vzhledem k odlišnosti šifrování se používají klíče přibližně 10x delší než u šifrování s tajným klíčem – klíče délky 512 bitů nejsou bezpečné, rozumnou úroveň bezpečnosti a výhled do budoucna mají klíče s délkou okolo 1000 bitů a více.

14.1 Elektronický podpis

Systémy šifrování s veřejným klíčem umožňují použít k šifrování soukromý klíč a k dešifrování klíč veřejný. Tento zdánlivě nesmyslný postup (zprávu může rozluštit kdokoli, kdo má veřejný klíč, čili v podstatě každý) lze využít k ověřování autenticity. Za předpokladu, že soukromým klíčem disponuje skutečně jen jeho vlastník, je zřejmé, že dokument, který lze dešifrovat určitým veřejným klíčem, zašifroval majitel příslušného soukromého klíče.

Elektronický podpis je kontrolní součet zprávy (a dalších dat, jako identifikace podepisující osoby a data a času podpisu) vytvořený speciální hašovací funkcí a poté zašifrovaný soukromým klíčem podepisující osoby.

14.1.1 Hašovací funkce

Hašovací funkce pro účely kryptografie je funkce, která slouží k výpočtu kontrolního součtu zprávy, a přitom má tu vlastnost, že je výpočetně nepřekonatelně složité vytvořit jinou smysluplnou zprávu, která má stejný součet (tj. podvrženou zprávu, která se tváří jako pravá).

Hašovací funkce lze používat kromě elektronických podpisů i k testování neporušenosti souborů, k rychlému porovnávání souborů (porovnávají se pouze hodnoty vrácené hašovací funkcí), k bezpečnému ukládání hesel a k autentizaci například při přihlašování na počítače.

14.2 Certifikáty

Distribuce veřejných klíčů je snadná. Problémem je naopak zajištění jejich autenticity (tj. prokázání, že určitý klíč skutečně patří osobě, která je uvedena jako jeho vlastník).

Technicky lze tento problém vyřešit tak, že odesilatel předá svůj klíč osobě, které příjemce důvěřuje, a tato osoba stvrdí svým elektronickým podpisem autenticitu klíče.

Toto potvrzení se nazývá certifikát.

Certifikát obsahuje kromě veřejného klíče informace pro koho byl vytvořený, kdy byl vytvořený, dokdy je platný, která certifikační autorita jej vydala a pro jaký účel.

Certifikační autorita je veřejně důvěryhodná instituce, která se zabývá vydáváním certifikátů. Aby její činnost měla právní sílu, musí být její fungování je řízeno zákonem.

14.2.1 Kvalifikovaný certifikát

Je certifikát vázaný na konkrétní fyzickou osobu.

14.3 Časová razítka

Časové razítko je elektronický dokument vydávaný například certifikační autoritou obsahující podpis jiného dokumentu a uvedení aktuálního času. Znemožňuje antedatování dokumentu.

14.4 Poznámky k praktické realizaci

Vzhledem k tomu, že šifrování veřejným klíčem je výpočetně velmi náročné a proto pomalé, v praxi se kombinuje se šifrováním s tajným klíčem. Pro zašifrování zprávy se vygeneruje náhodný tajný klíč a zpráva se jím zašifruje. Tento tajný klíč se zašifruje veřejným klíčem příjemce a přiloží k zašifrované zprávě.