Procesor

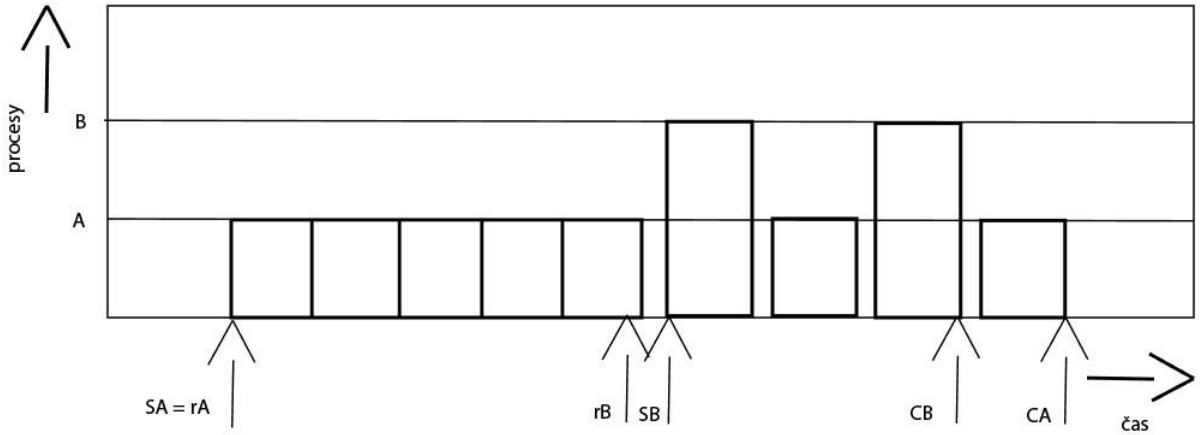
1. prinesie inštrukciu pamäte
2. dekóduje inštrukciu
3. prinesie operandy
4. vykoná inštrukcie
5. zapíše výsledky a nastaví príkazy

Operačný systém

- riadi procesy
- riadi pamäte
- riadi periférne zariadenia
- riadi súbory

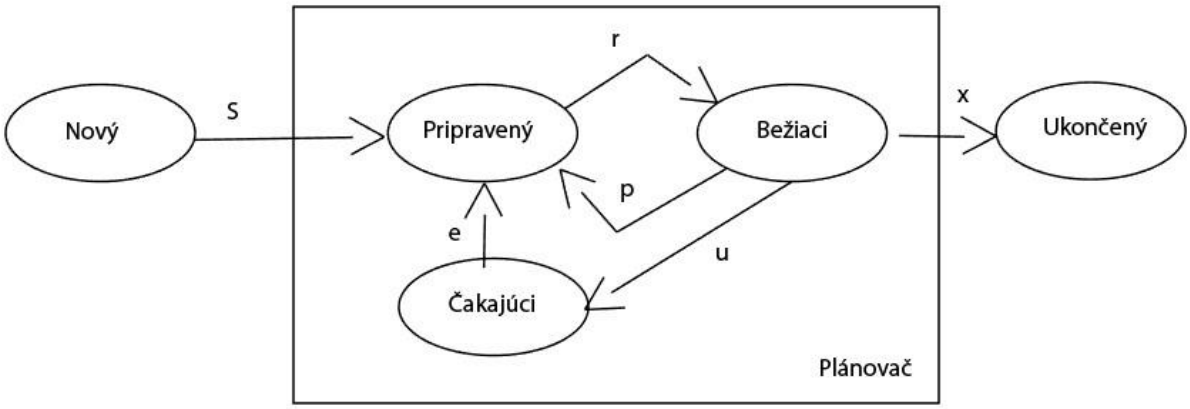
Riadenie procesorov

Multitasking

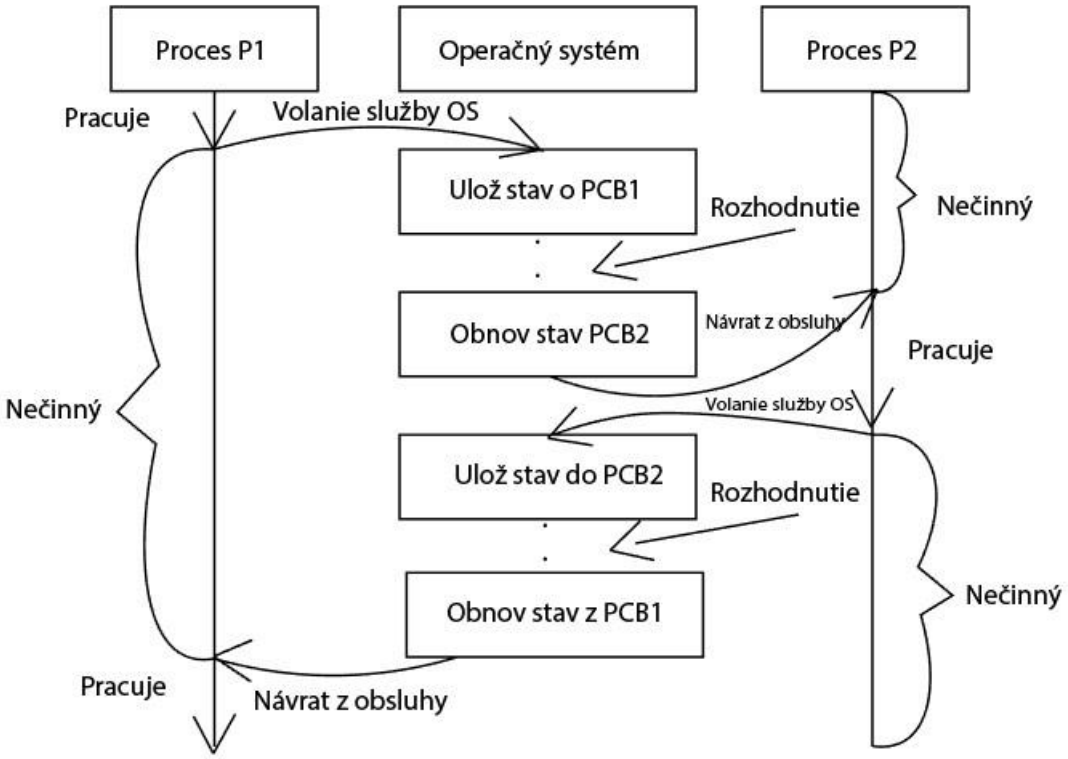


ri- doba príchodu (dostupnosti) procesu 2
Si – doba 1. spustenia (štartu procesu
2) Ci – doba dokončenia procesu 2 pi –
proceesing time pA = CA – SA pi = Ci –
Si
pre j procesorov → pij = Cij – Sij

Životný cyklus procesu



Process control blok (blok riadenia procesu)



Procesy a vlákna

Kód Dáta Súbor Registre

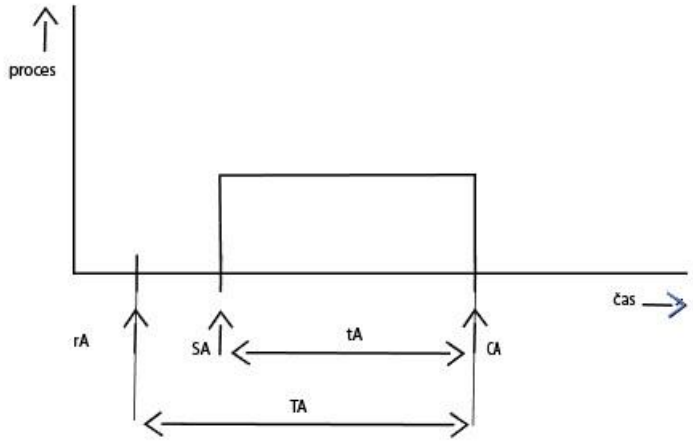
Zásobník vlákna

~

Plánovanie procesov

Plánovacie algoritmy

Určujú ako sú zoradené usporiadané procesy v rade prípravných procesov.



ti – požadovaná doba obsluhy

TA – celková doba obehu procesu i v systéme

Ri – proces zhoršenia kvality obehu procesu i

Proces	**ri** (príchod)	**ti** (pož. doba)	**Si** (štrukt)	**Ci** (dokon)	**Ti** (=CA-ri) (celk. oba)	**Ri** = $\frac{Ti}{ti}$
A	0	4	0	4	4	1.0
B	2	7	4	11	9	1.2
C	4	2	11	13	9	4.5
D	6	1	13	14	8	8.0
E	8	10	14	24	16	1.6

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
A	A	A	**A**	B	B	B	B	B	B	**B**	C	**C**	**D**	E	E	E	E	E	E	E	E	E	E	E

Algoritmy

FCFS – First Come

- spracovanie v poradí príchodu
- bez preempcie

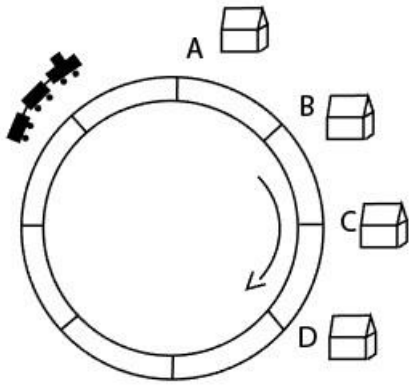
Round Robin

- spracovanie v poradí príchodu

- s preempcie

Proces	ri (príchod)	ti (pož. doba)	Si (štrukt)	Ci (dokon)	Ti (=CA-ri) (celk. oba)	Ri = $\frac{Ti}{ti}$
A	0	4	0	8	8	2.0
B	2	7	2	18	16	2.2
C	4	2	5	10	6	3.0
D	6	1	6	7	1	1.0
E	8	10	10	24	17	1.6

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
A	A	A	**A**	B	B	B	B	B	B	**B**	C	**C**	**D**	E	E	E	E	E	E	E	E	E	E	E



4 kritéria kvality systému

1. spravodlivosť
2. minimálna odozva
3. maximálna priepustnosť – zodpovedá, počet okam. procesov za jednotku času
4. minimálna doba obehu procesu i Ti

3. Najkratší proces najskôr

Short process next SPN bez preempcie –
nedostatok „starvácia procesu“

Proces	ri (príchod)	ti (pož. doba)	Si (štrukt)	Ci (dokon)	Ti (=CA-ri) (celk. oba)	Pi = $\frac{Ti}{ti}$
A	0	4	0	4	4	1.0
B	2	7	7	14	12	12/7
C	4	2	4	6	2	1.0
D	6	1	6	7	1	1.0
E	8	10	14	24	16	1.6

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
A	A	A	**A**	C	C	D	B	B	B	**B**	B	**B**	**B**	E	E	E	E	E	E	E	E	E	E	

4. Najkratší proces najskôr s preempciou

Preempt Short Process Next – PSPN s preempciou – čiastočne odstranená

- má vplyv na zhor. kval.

Proces	ri (príchod)	ti (pož. doba)	Si (štrukt)	Ci (dokon)	Ti (=CA-ri) (celk. oba)	Pi = $\frac{Ti}{ti}$
A	0	4	0	4	4	1.0
B	2	7	7	14	12	12/7
C	4	2	4	6	2	1.0
D	6	1	6	7	1	1.0
E	8	10	14	24	16	1.6

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
A	A	A	**A**	C	C	D	B	B	B	**B**	B	**B**	**B**	E	E	E	E	E	E	E	E	E	E	

Proces	ri (príchod)	ti (pož. doba)	Si (štrukt)	Ci (dokon)	Ti (=CA-ri) (celk. oba)	Pi = $\frac{Ti}{ti}$

A	0	10				
B	2	4				
C	4	1				
D	6	7				
E	8	2				

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
A	A	B	**B**	C	B	B	D	E	E	**D**	D	**D**	**D**	D	D	A	A	A	A	A	A	A	A	

5. Algoritmus podľa penalizačnej fcie; pi = 0 po

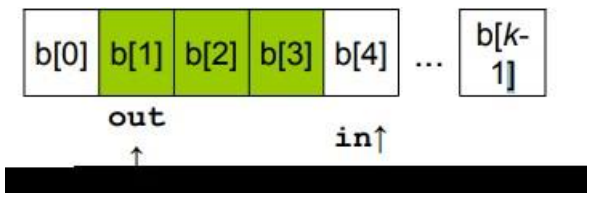
každom uplynutí pi ← pi+1

S PREM (RR +PF)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
A	A	B	**A**	B	C	B	B	D	E	**A**	E	**A**	**D**	A	D	A	D	A	D	A	D	A	D	
0	0	0	**0**	0	0	0	0	0	0	**1**	1	**1**	**1**	1	1	1	1	1	1	2	2	2	2	
		0	**0**	0	0	1	1	1	1	**2**	2	**2**	**2**	3	3	3	3							
				0	1	2	3	4	5	**6**	7													
						0	0	0	0	**0**	0	**0**	**1**	1	1	1	1	1	1	2	2	2	2	
								0	0	**1**	1	**2**	**2**	3	3	4	4	5	5	6	6			

Synch. procesov

- Producent – zapisuje (generuj do vyrovnanej pamäti s konečnou kapacitou) • Konzument – vyberie správu a ďalej s ňou pracuje



- in – koľko je zapísaných
- out – číta sa
- in + 1 – pôjde od začiatku alebo zvýš o 1
- wh = 0 nemôže pracovať

Producent:

```
repeat
        {vytvor dalsiu do nova_hodnota}

        while citac = n do  nic_nerob;
        buffer(in) := nova_hodnota;
        in := in + 1 mod n;  citac
        := citac + 1;
until false;
```

Konzument:

```
repeat  while citac = 0 do
        nic_nerob;
        prijata_hodnota := buffer(out);
        out := out + 1 mod n;  citac :=
        citac - 1;

        {zpracuj prijatu v prijata_hodnota}  until
false;
```

Problém súperenia (*race condition*)

- inkrementácia citac bude implementované asi takto:

P_1:	register$_1$	← citac	`move citac,D0`
P_2:	registr$_1$	← register$_1$ + 1	`add D0,#1`
P_3:	citac	← register$_1$	`move D0,citac`

- dekrementácia citac bude zrejme implementované ako:

K_1:	register$_2$	← citac	`move citac,D4`
K_2:	register$_2$	← register$_2$ – 1	`sub D4,#1`
K_3:	citac	← register$_2$	`move D4,citac`

- Tam, kde platia Murphyho zákony, **môže** nastať nasledujúca postupnosť prekladania producenta a konzumenta (na počiatku citac = 3)

Interval	Beží	Akcia	Výsledok
P_1	producent	register$_1$ ← citac	register$_1$ = 3
P_2	producent	register$_1$ ← register$_1$ + 1	register$_1$ = 4
K_1	konzument	register$_2$ ← citac	register$_2$ = 3
K_2	konzument	register$_2$ ← register$_2$ – 1	register$_2$ = 2
P_3	producent	citac ← register$_1$	citac = 4
K_3	konzument	citac ← register$_2$	citac = 2

- Na konci je citac = 2 alebo 4, ale programátor chcel mať 3
- Je to dôsledkom **nepredvídateľného** prekladania procesov vplyvom možnej preempcie

Operačné systémy I.

Kritická sekcia – časť programu, ktorá používa spol. premenné

Požiadavky pre riešenie problému kritických sekcií

1. **Vzájomné vylúčenie** – podmienka bezpečnosti (Mutual Exclusion)
   - len jeden môže byť
   - pokiaľ proces P_i je vo svojej kritickej sekcii, potom žiadny ďalší proces nesmie byť vo svojej kritickej sekcii združenej s rovnakým prostriedkom
2. **Trvalosť postupu** – podmienka živosti (Progress)
   - ak nie je krit. obl. nemôže obmedzovať

- ak žiadny proces nevykonáva svoju kritickú sekciu združenú s istým zdrojom a existuje aspoň jeden proces, ktorý si želá vstúpiť do kritickej sekcie združenej s týmto zdrojom, potom výber procesu, ktorý do takejto kritickej sekcie vstúpi, sa nesmie odkladať nekonečne dlho.

3. **Konečné čakanie** – podmienka spravodlivosti (Fairness) –
    - žiaden proces nemôže nekon. dlho čakať na vstup do krit. oblasti
    - proces smie čakať na povolenie vstupu do kritické sekcie len konečnú dobu.
    - musí existovať obmedzenie počtu, koľkokrát môže byť povolený vstup do kritické sekcie združenej s istým prostriedkom iným procesom než procesu požadujúcemu vstup v dobe medzi vydaním žiadosti a jej uspokojením.

Možnosti riešenia problému kritických sekcií

- Základná štruktúra procesu s kritickou sekciou vstupná sekcia
    - kritická sekcia ukončovacia sekcia z
- ostávajúca sekcia until false
- Implementácia **vstupnej sekcie** a **ukončovacej sekcie** je kľúčom k riešeniu celého problému kritických sekcií.
- Čisto softvérové riešenie na aplikačnej úrovni (pokiaľ ste aktívny –celé zle) ○ Algoritmy, ktorých správnosť sa nespolieha na žiadnu ďalšiu podporu ○ Základné riešenie s **aktívnym čakaním** (busy waiting)
- Hardvérové riešenie ○ Pomocou špeciálnych inštrukcií procesoru ○ Stále ešte **s aktívnym čakaním**
- Softvérové riešenie sprostredkované operačným systémom ○ Potrebné služby a dátové štruktúry poskytuje OS (napr. **semafory**) ○ Tým je umožnené **pasívne čakanie** – proces nesúťaží o procesor ○ Podpora volania služieb v programovacích systémoch/jazykoch (napr. **monitory, zasielanie správ**)

Vzájomné vylúčenie s aktívnym čakaním

- Zamykacie premenné ○ Kritickú sekciu „ochránime" **zdieľanou zamykacou premennou** združenou so zdieľaným prostriedkom.
    - Pred vstupom do kritickej sekcie proces testuje túto premennú a ak je nulová, nastaví ju na 1 a vstúpi do kritickej sekcie. Ak nemá premenná hodnotu 0, proces čaká v cykle (**aktívne čakanie** – busy waiting).
      while lock <> 0 do nic_nerob;
    - Pri opustení kritickej sekcie proces túto premennú opäť nuluje. lock := 0;
    - **Čo sme dosiahli**? Nevyriešili sme nič: súbeh sme preniesli na zamykaciu premennú ○ Myšlienka zamykacích premenných však nie je úplne chybná

Semafór

- Block – ak ide niečo do krit. všetko zablokuje
- Odblock – odblokuje

- Semafory sú synchronizačné prostriedky, ktoré navrhol E.W.Dijkstra.
- Používajú pasívne čakanie procesov na uvoľnenie zdieľaného prostriedku a dovoľujú chrániť naraz viac prostriedkov určitého druhu.

- Môžeme ich použiť aj pre riadenie práce periférneho zariadenia.
- Dovoľujú korektne zabezpečiť vzájomné vylučovanie medzi ľubovoľným počtom procesov.
- Pritom sa vzájomne vylučujú len procesy chránené tým istým semaforom.

Použijeme tri semafory:

- Semafor **Prazdna** bude strážiť počet voľných miest vyrovnávacej pamäti, do ktorej zapisuje **Výrobca**.
- Semafor **Plna** bude strážiť počet zaplnených miest vyrovnávacej pamäte, z ktorých číta **Spotrebiteľ**.
- Semafor **Volno** bude strážiť výlučný prístup k vyrovnávacej pamäti, ktorá je zdieľaným prostriedkom.

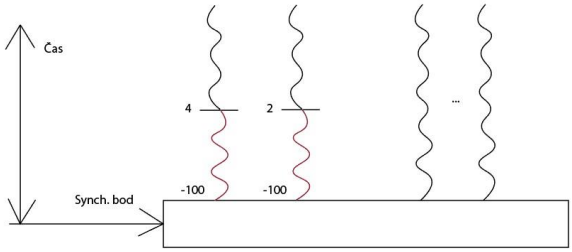
Príklad : 2x zapíše, 2x prečíta:

Semaforovská premenná		**1. zápis**		**2. zápis**		**1. čítanie**		**2. čítanie**	
Prazdna.Pocet	**10**	**9**		**8**			**9**		**10**
Plna.Pocet	**0**		**1**		**2**	**1**		**0**	
Volno.Pocet	**1**	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**1**

```
Procedure Wait(var S:semaphore);
 begin
   VstupOblasti(S.MutEx);
   if (S.Pocet=0) then   Block(S.Rad);
{PROCES BLOKOVANÝ}
    S.Pocet:=S.Pocet-1;
   KoniecOblasti(S.MutEx);
     end;
```

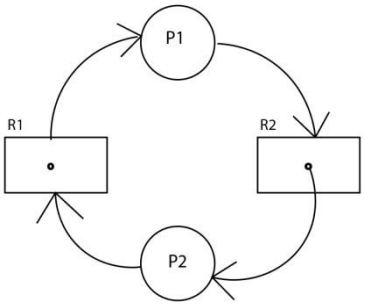
```
Procedure Signal(var S:semaphore);
 begin
    VstupOblasti(S.MutEx);
      S.Pocet:=S.Pocet+1;
       if (S.Pocet=1) then  dBlock(S.Rad);
    KoniecOblasti(S.MutEx);
    end;
```

Uviaznutie



- Zo života: Ak čakám na kamaráta pred domom, kt. mešká. Počkám a prídeme neskoro do školy.
- strac. času: - 1. krát idem do obchodu na nákup (stratím veľa času
  - 2. krát – triedený zoznam (drogéria, kuchynka,..., menej času) **Graf**

pridel. zdrojov



Uviaznutie procesov – deadlock

Uviaznutie – zázračný jav, len pri paral.

Získanie zdroja

Vlák A Vlák B – pýtajú si rozdieln reso.

ac – pýta
ul – uvoľní

Charakteristika uviaznutia

Coffman formuloval štyri podmienky, ktoré musia platiť súčasne, aby uviaznutie mohlo vzniknúť

1. **Vzájomné vylúčenie**, Mutual Exclusion
   - zdieľaný zdroj môže v **jednom** okamihu používať **najviac jeden proces**
2. **Postupné uplatňovanie požiadaviek**, Hold and Wait
   - proces vlastniaci aspoň jeden zdroj **potrebuje ďalší**, ale ten je vlastnený iným procesom, v dôsledku čoho bude **čakať na jeho uvoľnenie**
3. **Nepripúšťa sa odnímanie zdrojov**, No preemption
   - zdroj môže uvoľniť iba proces, ktorý ho vlastní, a to **dobrovoľne**, keď už zdroj nepotrebuje

4. **Zacyklenie požiadaviek**, Circular wait
    - existuje **množina čakajúcich procesov** {P_0 , P_1 , ... , P_k , P_0 } takých, že P_0 čaká na uvoľnenie zdroja držaného P_1 , P_1 čaká na uvoľnenie zdroja držaného P_2 , ... , P_{k-1} čaká na uvoľnenie zdroja držaného P_k , a P_k čaká na uvoľnenie zdroja držaného P_0 .
    - v prípade **jednoinštančných** zdrojov splnenie tejto podmienky značí, že k **uviaznutia už došlo**

Množina procesov uviazla, keď každý proces z množiny čaká na uvoľnenie prostriedku.

Algoritmus bankára

- hľadáme, také poradie spracovania procesov, aby bol stav bezpečný.
- banka príjma peniaze, požičiava

    Stred čo potrebuje, Prvý banka dala Potrebné

    dopočítavame

	Umiestnené				Proces max. žiada				Potrebné				Dostupné (rezerva)		
	A	B	C		A	B	C		A	B	C		A	B	C
P0	0	1	0		7	5	3		7	4	3		3	3	2
P1	2	0	0		3	2	2		1	2	2				
P2	3	0	2		9	0	2		6	0	0				
P3	2	1	1		2	2	2		0	1	1				
P4	0	0	2		4	3	3		4	3	1				

	Umiestnené				Proces max. žiada				Potrebné				Dostupné (rezerva)		
	A	B	C		A	B	C		A	B	C		A	B	C
~~P0~~	~~0~~	~~1~~	~~0~~		~~7~~	~~5~~	~~3~~		~~7~~	~~4~~	~~3~~		~~3~~	~~3~~	~~2~~
~~P1~~	~~2~~	~~0~~	~~0~~		~~3~~	~~2~~	~~2~~		~~1~~	~~2~~	~~2~~				
~~P2~~	~~3~~	~~0~~	~~2~~		~~9~~	~~0~~	~~2~~		~~6~~	~~0~~	~~0~~				
~~P3~~	~~2~~	~~1~~	~~1~~		~~2~~	~~2~~	~~2~~		~~0~~	~~1~~	~~1~~				
P4	0	0	2		4	3	3		4	3	1				

	Umiestnené				Proces max. žiada				Potrebné				Dostupné (rezerva)		
	A	B	C		A	B	C		A	B	C		A	B	C
P1	2	0	0		3	2	2		1	2	2		3	3	2
P1	3	2	2		3	2	2		1	2	2		2	1	0
P1	0	0	0		~~3~~	~~2~~	~~2~~						5	3	2

	Umiestnené				Proces max. žiada				Potrebné				Dostupné (rezerva)		
	A	B	C		A	B	C		A	B	C		A	B	C
P3	2	1	1		2	2	2		0	1	1		5	3	2
P3	2	2	2		2	2	2		0	1	1		5	2	1
P3	0	0	0		~~2~~	~~2~~	~~2~~						7	4	3

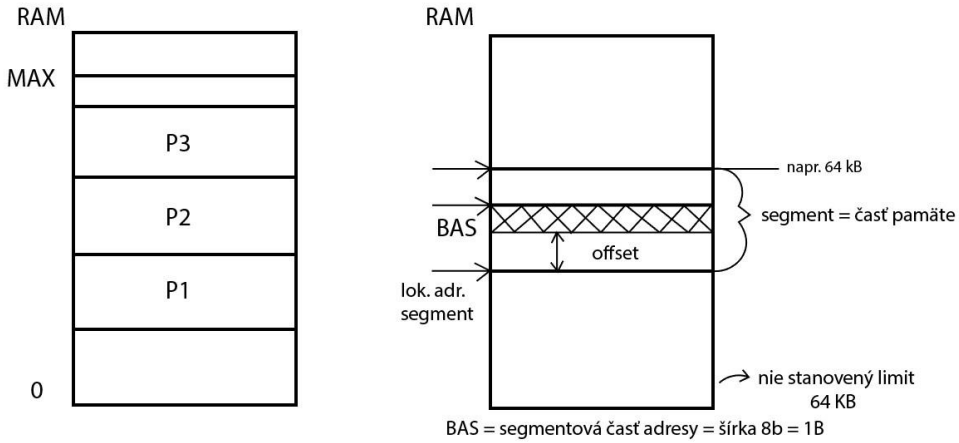
	Umiestnené				Proces max. žiada				Potrebné				Dostupné (rezerva)		
	A	B	C		A	B	C		A	B	C		A	B	C
P0	0	1	0		7	5	3		7	4	3		7	4	3
P0	7	5	3		7	5	3		7	4	3		0	0	0
P0	0	0	0		~~7~~	~~5~~	~~3~~						7	5	3

	Umiestnené				Proces max. žiada				Potrebné				Dostupné (rezerva)		
	A	B	C		A	B	C		A	B	C		A	B	C
P2	3	0	2		9	0	2		6	0	0		7	5	3
P2	9	0	2		9	0	2		6	0	0		1	5	3
P2	0	0	0		~~9~~	~~0~~	~~2~~		6	0	0		10	5	5

	Umiestnené				Proces max. žiada				Potrebné				Dostupné (rezerva)		
	A	B	C		A	B	C		A	B	C		A	B	C
P4	0	0	2		4	3	3		4	3	1		10	5	5
P4	4	3	3		4	3	3		4	3	1		6	2	4
P4	0	0	0		4	3	3		4	3	1		10	5	7

Riadenie pamäte

RAM – aký rýchly bude PC (čím viac, tým lepší)



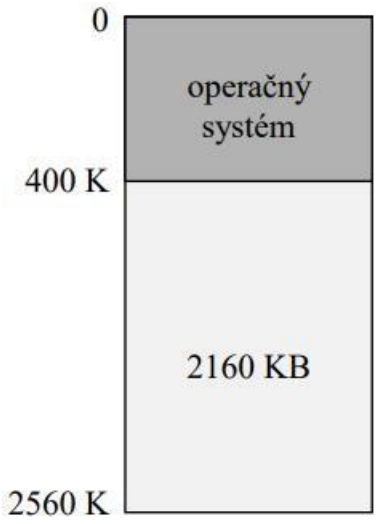
Logická (x x x x h) adresa segment: ofset

BAS – fyzická adresa **x x x x** 0 h <- končí nulou

BAS:	x x x x 0	napr.: 020A0
	+ ofset	1BCD
	----------	--------
	2	03C6D

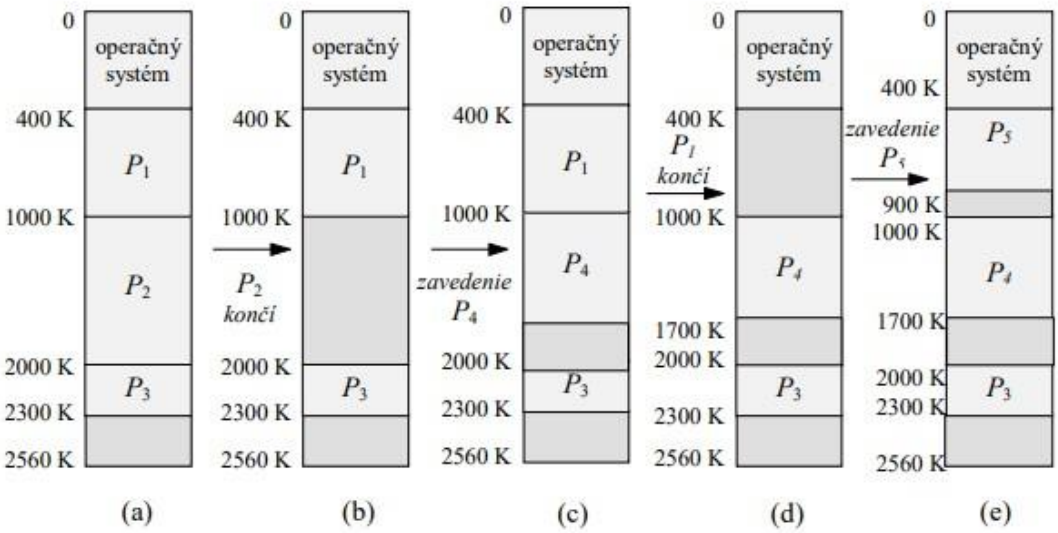
Segmentová adresa je uložená v segm. registri **procesora**.

- Tieto registre sa volajú **segmentové registre**
  - kódové
  - dátové
  - zásobníkové
  - extraddátové



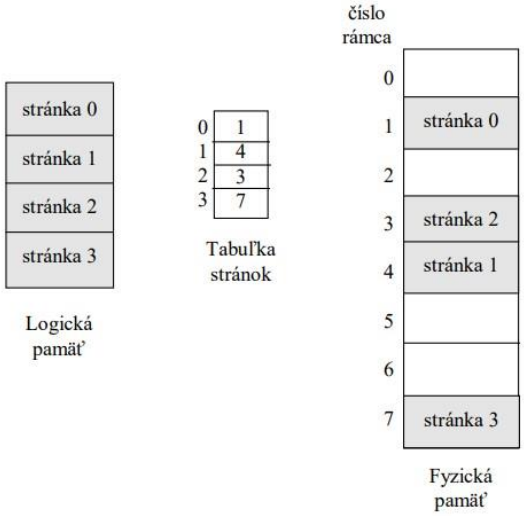
front pripravených procesov

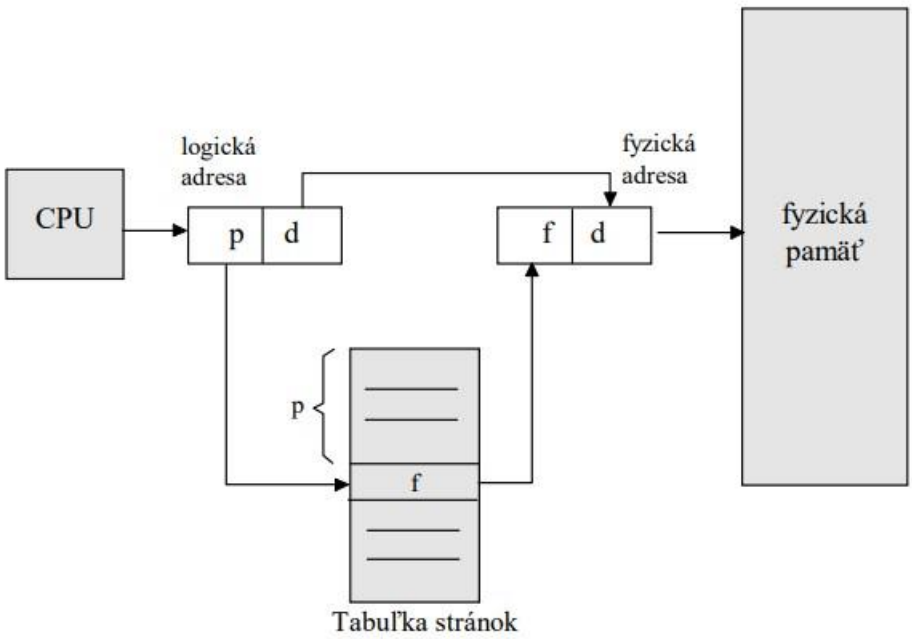
proces	požadovaná pamäť	požadovaný čas
P_1	600KB	10
P_2	1000KB	5
P_3	300KB	20
P_4	700KB	8
P_5	500KB	15

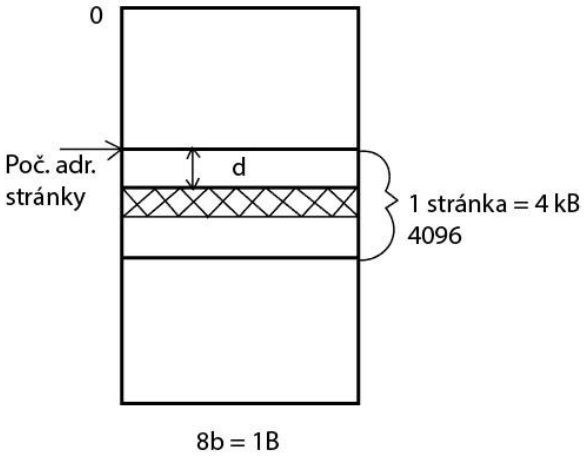


Stránkovanie

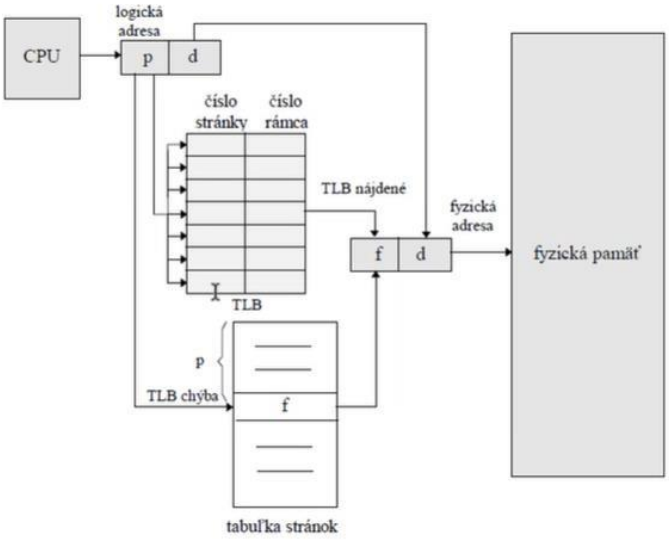
Stránky



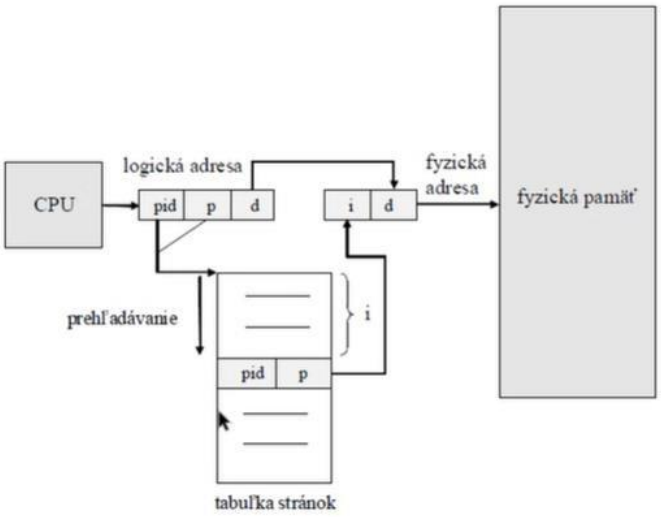




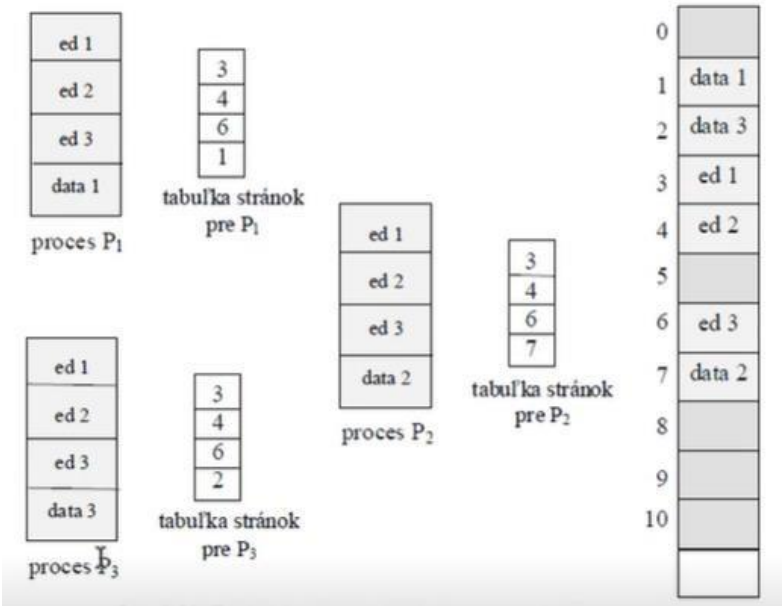
Pokračovanie

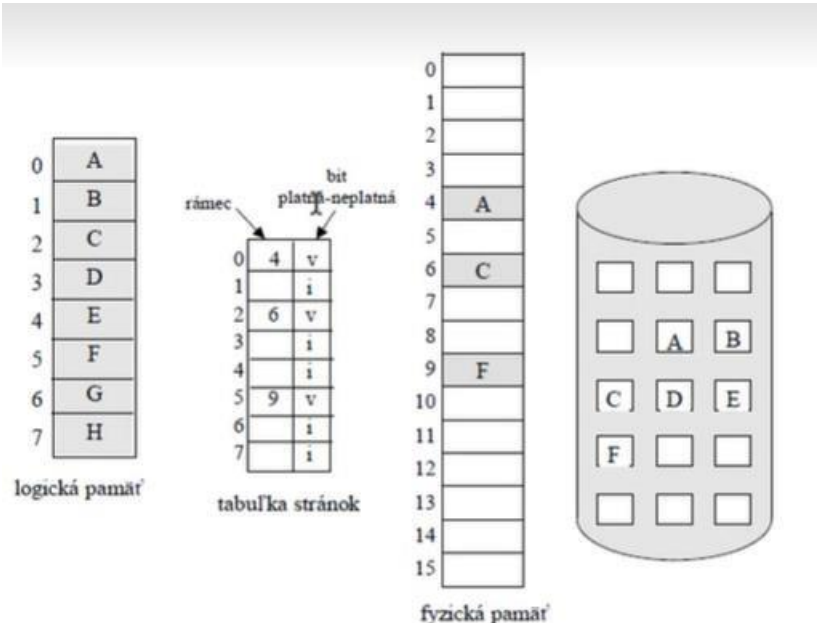


Invertovaná tabuľka stránok (vyhľadávame podľa čísla procesu)

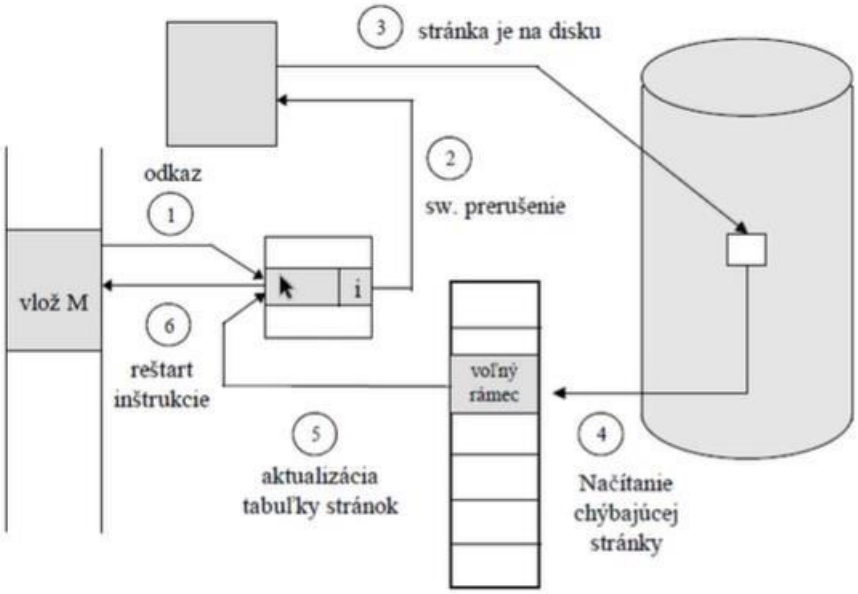


Zdieľanie stránok


ed 1
ed 2
ed 3
data 1
proces P1
3
4
6
1
tabuľka stránok pre P1
ed 1
ed 2
ed 3
data 2
proces P2
3
4
6
7
tabuľka stránok pre P2
ed 1
ed 2
ed 3
data 3
proces P3
3
4
6
2
tabuľka stránok pre P3
0
1
2
3
4
5
6
7
8
9
10
data 1
data 3
ed 1
ed 2
ed 3
data 2



Výpadky



Hľadáme taký algoritmus, ktorý nám odsunie takú stránku ktorá nám už nebude treba.

Algoritmus FIFO je jednoduchý.

Jeho výkonnosť, ale nie je vždy dobrá.

Nahradená stránka môže obsahovať inicializačný modul, ktorý už nebude potrebný, ale tiež to môže byť často používaná stránka, ktorá ihneď po odsunutí spôsobí nový výpadok stránky.

	7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
rámec 1	7	7	7	2		2	2	4	4	4	0			0	0			7	7	7
rámec 2		0	0	0		3	3	3	2	2	2			1	1			1	0	0
rámec 3			1	1		1	0	0	0	3	3			3	2			2	2	1

celkom 15 výpadkov

Obr. 9.8 Nahradzovací algoritmus FIFO

Algoritmus FIFO pri výmene stránky, ktorá je často používaná, môže spôsobiť zvýšenie počtu výpadkov.

Optimálny nahrádzací algoritmus (neodsunúť takú stránku, ktorá mi je treba)

	7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
r1																				
r2																				
r3																				

	7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
r1	7	7	7	2		2		2			2			2				7		
r2		0	0	0		0		4			0			0				0		
r3			1	1		3		3			3			1				1		

FIFO

	7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
r1																				
r2																				
r3																				

	7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
r1	7	7	7	2		2	2	4	4	4	0			0	0			7	7	7
r2		0	0	0		3	3	3	2	2	2			1	1			1	0	0
r3			1	1		1	0	0	0	3	3			3	2			2	2	1

Nahrádzací algoritmus LRU – Najdlhšie nepoužitá stránka

	7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
r1																				
r2																				
r3																				

	7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
r1	7	7	7	2		2		4	4	4	0			1		1		1		
r2		0	0	0		0		0	0	3	3			3		0		0		
r3			1	1		3		3	2	2	2			2		2		7		

Počet výpadkov: 7

Riadenie súborov

(FAT – file allocation table)

- Súbor je zoskupenie diskrétnych údajov - súbor údajov (dát). Ukladá sa do pamäťového zariadenia (disk).
- Súborový systém určuje spôsob ukladania a organizovania súborov a údajov tak, aby k nim bol umožnený jednoduchý prístup.

Súborové systémy (File systems)

1. MSDOC (diskový operačný systém)
2. NTFS (new technology
3. EXT

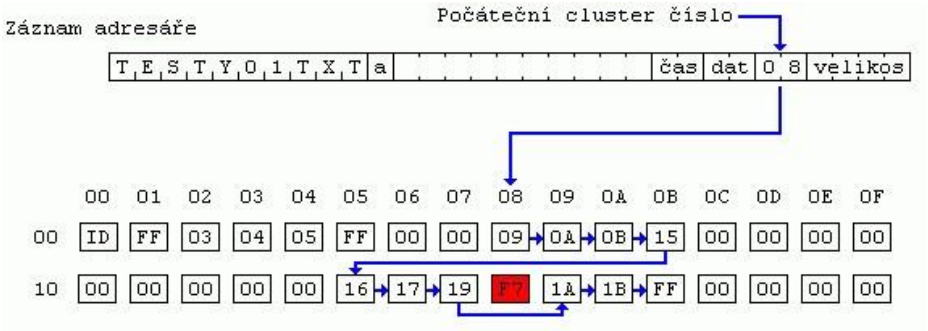
1. MSDOC

Štruktúra na disku:

Boot sektor FAT – 1. kópia FAT – 2. kópia Hlavný adresár disku Oblasť údajov – obsahuje údajov	MBR 1. partícia	2. partícia

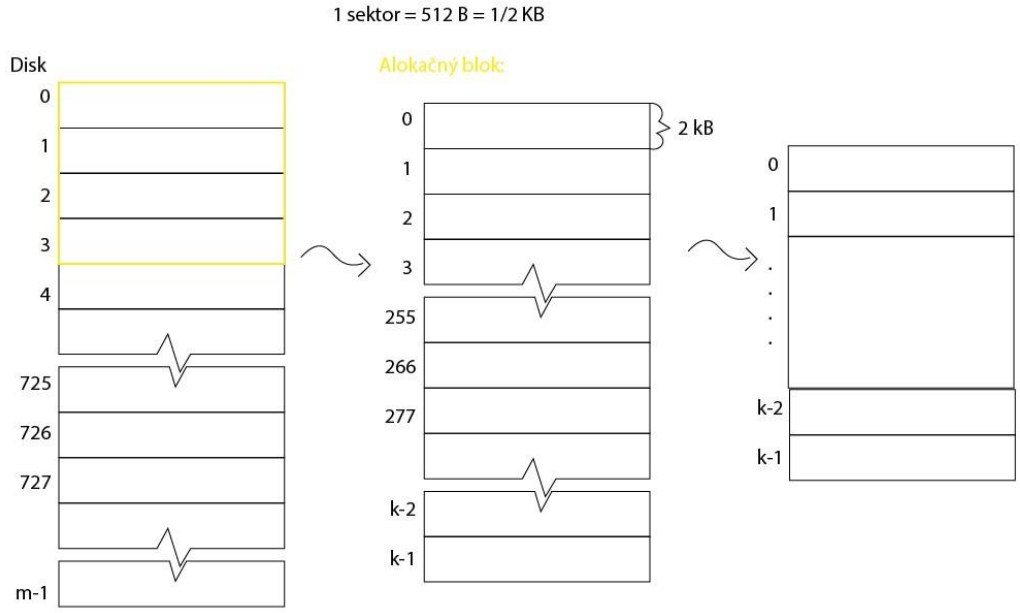
Boot sektor	FAT – 1. kópia	FAT – 2. kópia	Hlavný adresár disku Root	Oblasť údajov – obsahuje bloky údajov

FAT – tabuľka umiestnenia súborov (angl. File allocation table)



(08 je najdôl. – ukazuje tam kde je ulož. ...)

ROOT

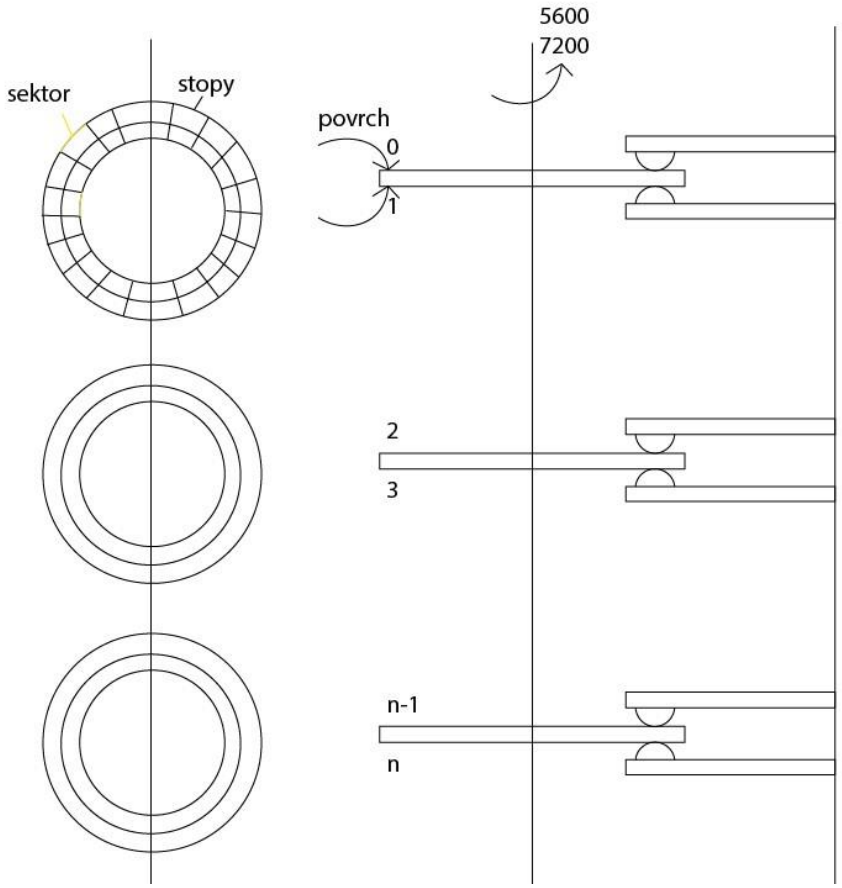


FAT:
(má toľko položiek koľko má alokačných blokov)

Obratom je ukazovateľ na ďalšiu položku

08, 09, 0A, 0B, 15, 16, 17, 18, 19

1A, 1B FF → konc. znak



Operačný systém

- Riadenie procesov
- Riadenie súborov
- Riadenie pamäti
- Riadenie periférnych zariadení

Správa periférnych zariadení

- PC komunikuje s prostredím pomocou periférnych zariadení

Klasifikácia vstupno/ výstupných zariadení

- zdieľateľné – môže použ. viac procesorov naraz **stav** sa dá ľahko **zapamätať**
- nezdielateľné – môžu slúž. viac zar., **stav** takéhoto zar. sa nedá obnoviť

Podľa spôs. prenosu môžeme rozdeliť:

- blokové – ukl. ino. po blokoch • znakové – prac s postupnosťou znakov
- iné zar.

2 komponenty – štruktúraa V/V zar.

- elektronický komponent – naz. sa **radič**, **adaptér** (HW)
- mechanický –

OS takmer vždy komunikuje s radičom

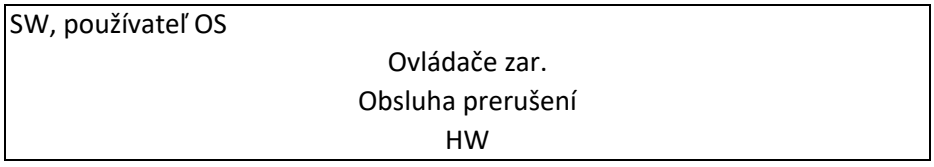
Niektoré PC používajú **kanály** – preberajú riadenie zar.

Radič má niekoľko **registrov**, použ. pri komun. s **procesorom**

- stavový
- príkazový
- dátový
- adresný

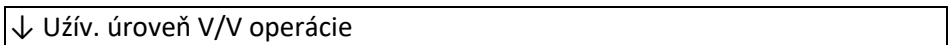
V/V operácia prebieha:

1. OS odovzdá príkazy do **ovládača** a tam zostavuj **registre radiča**
2. Po prebr. príkazu radičom sa proc. môže venovať inej práci.
3. Keď perif. skon. radič to oznámi OS pomocou prerušenia
4. Proc zistí výsledky vykon. práce z registrov radiča



Radič zabezpečí prečitan. bloku, skontr. prenos, vyvolá prerušenie

Vrstvy prog. vytvorenia pre obsluhu V/V zar a ich funkcie



↓ Progr. vybavenie, nezávislé od zariadenia	↑
↓ Ovládače zariadení	↑
Obsluha	↑
HW	↑

- vylúč. prid. zar.
- virtualizácia nezdieľ. zar **Spooling** – technika pri správe tlačiarní

Organ. prg. vybavenia pre správu perif. zar.

1. Operácie na užívateľskej úrovni

- operácie so zariedeniami
- operácie pre prác. so súbormi

Operácie so zar.:

- Žiadosť
- Čítanie
- Zistenie a nastavenie
- Logic. pripojenie

Operácie pre prácu so súb.:

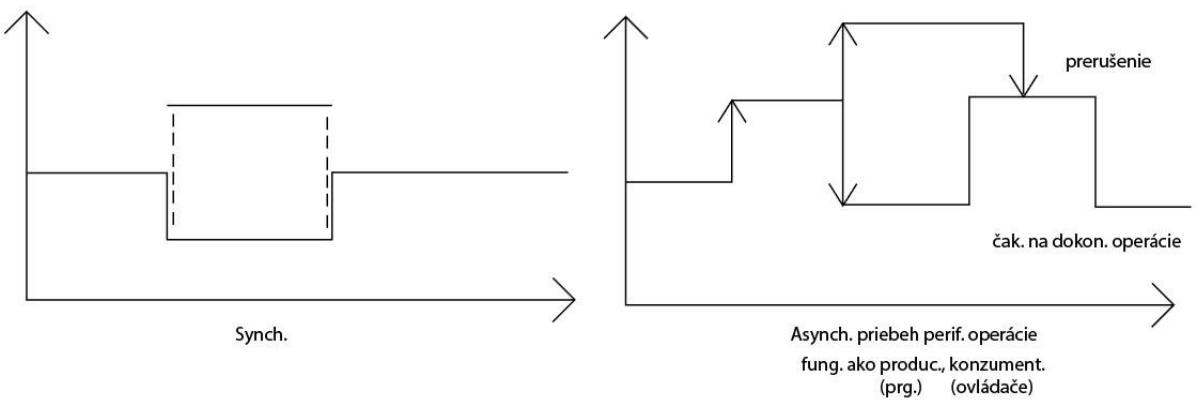
- Tvorba
- Otvorenie
- Čítanie
- Zistenie a nastavenie

2. Ovl. perif. zar.

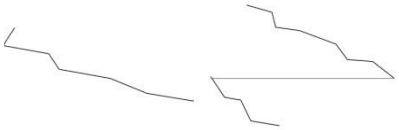
Ovládač je program kt. **preberá** od prg. **požiadavky na vykonanie** operácie a **odovzdáva ich zariadeniu.**

V ovládači sú zabud.

- Aké príkazy rešpektuje
- Údaje, kt. sú iné pre každý typ zariadenia
- konk. údaje o zariadení



Algoritmus výťahu – SCAN – pozbiera



- C-SCAN – smerom hore