Počítačové sítě a operační systémy

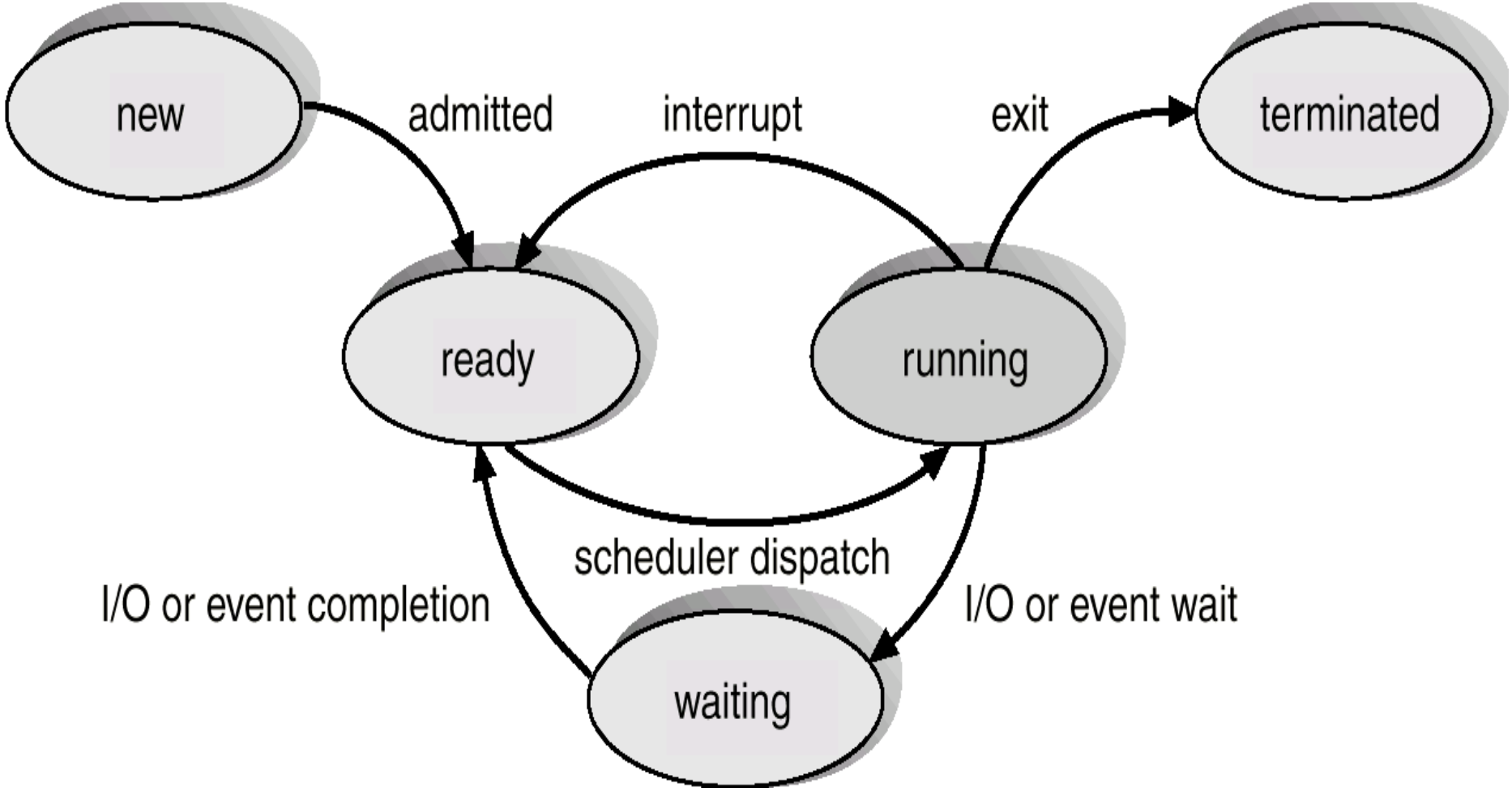
Plánování CPU
Správa paměti

Jaromír Plhák
xplhak@fi.muni.cz

Motivace k plánování

- V multitaskingových systémech existuje více procesů připravených k běhu
- Procesorů je ve výpočetním systému (prakticky vždy) méně než procesů
- OS musí rozhodovat, který proces poběží jako příští, tj. kterému procesu přidělí (případně na omezenou dobu) procesor

Stavy procesu



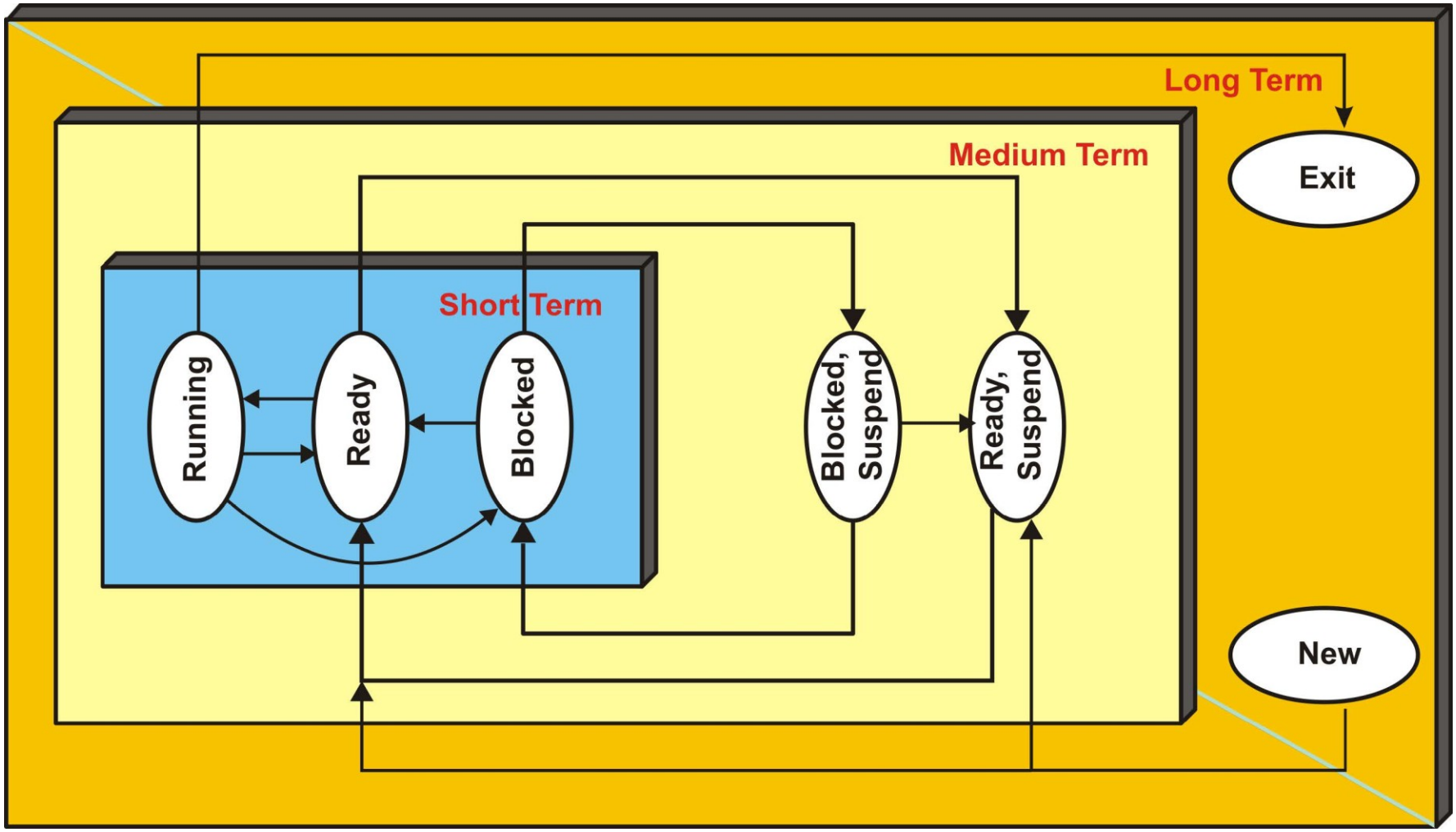
Plánovače v OS (1)

- Dlouhodobý plánovač (strategický plánovač, *job scheduler)*
  - Vybírá který požadavek na výpočet lze zařadit mezi procesy
  - Definuje stupeň multiprogramování
  - Je vyvoláván řídce, nemusí být rychlý
  - Nejlepšího výsledku dosáhneme při vhodné kombinaci procesů orientovaných na I/O a na využití CPU

Plánovače v OS (2)

- Střednědobý plánovač (taktický plánovač)
  - Logicky náleží částečně i do správy hlavní paměti (FAP)
  - Taktika využívání omezené kapacity FAP při multitaskingu
  - Vybírá který proces je nutné zařadit mezi odložené procesy (odebírá mu prostor ve FAP)
  - Vybírá kterému odloženému procesu lze opět přidělit prostor ve FAP
- Krátkodobý plánovač (operační plánovač, dispečer, *dispatcher)*

Plánovače v OS (3)



Kritéria kvality plánování (1)

- Uživatelsky orientovaná kritéria
  - Doba obrátky, *Turnaround time*
    - Doba běhu + doba čekání na zdroje vč. CPU
    - Vhodná míra pro dávkové zpracování
    - Přirozená je snaha o minimalizaci doby obrátky v dimenzi doby čekání
    - Normalizovaná doba obrátky = doba obrátky / doba běhu
  - Doba reakce, *Response time*
    - Míra pro interaktivní systémy
    - Doba od zadání požadavku do doby očekávané reakce
    - Cíl – snaha o minimalizaci pro co největší komunitu

Kritéria kvality plánování (2)

- Systémově orientovaná kritéria
  - Propustnost, *Throughput*
    - Počet procesů dokončených za jednotku času
    - Přirozená je snaha o maximalizaci propustnosti
    - Požadavek dosažení vysoké propustnosti nebývá kompatibilní s požadavkem minimalizace doby obrátky
  - Využití CPU
    - Maximalizace ve víceuživatelských systémech
    - Pro real-time systémy a jednouživatelské systémy nepodstatné kritérium
  - Spravedlivost, *Fairness*
    - Porovnatelné procesy musí získat porovnatelnou

Plánování CPU – Dispečer

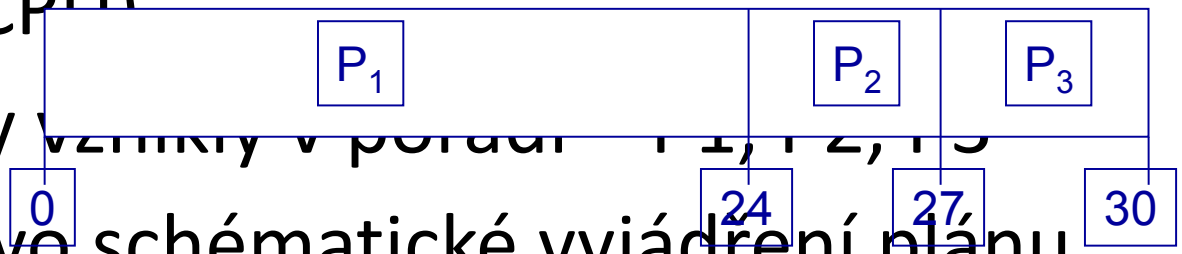
- Vybírá proces, kterému bude přidělen CPU
- Vybírá jeden z procesů, které jsou zavedeny operační paměti a které jsou „připravené“
- Plánovací rozhodnutí může vydat v okamžiku, kdy proces
  - 1. přechází ze stavu běžící do stavu čekající
  - 2. přechází ze stavu běžící do stavu připravený
  - 3. přechází ze stavu čekající do stavu připravený
  - 4. končí
- Případy 1 a 4 se označují jako nepreemptivní

Dispečer

- Předání procesu procesoru zahrnuje
  - Přepnutí kontextu
  - Přepnutí režimu procesoru na uživatelský režim
  - Skok na odpovídající místo v uživatelském programu pro opětovné pokračování v běhu procesu
- Dispečerské zpoždění (Dispatch latency)
  - Doba, kterou potřebuje dispečer pro pozastavení běhu jednoho procesu a start běhu jiného procesu

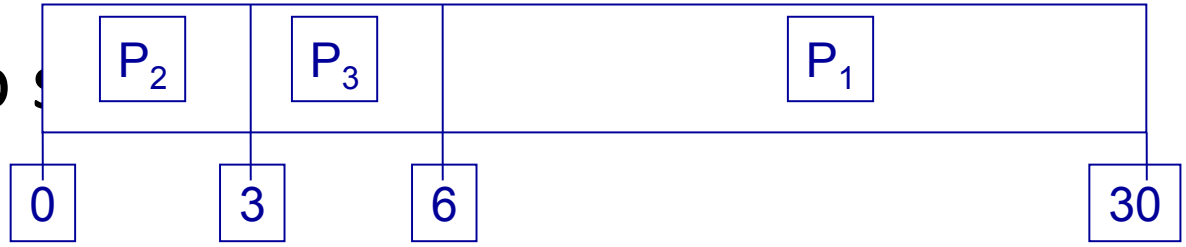
Algoritmus FCFS (1)

- Algoritmus „Kdo dřív přijde, ten dřív mele“ (First Come, First Served), FCFS
- Máme 3 procesy P1 (vyžaduje 24 dávek CPU), P2 (vyžaduje 3 dávky CPU), P3 (vyžaduje 3 dávky CPU)
- Procesy vznikly v pořadí P1, P2, P3
- Ganttovo schématické vyjádření plánu



Algoritmus FCFS (2)

- Varianta jiná – procesy vznikly v pořadí P2, P3, P1
- Ganttovo s



- Doby čekání – P2 = 0, P3 = 3, P1 = 6
- Průměrná doba čekání – (6+0+3)/3 = 3

Round Robin (RR) (1)

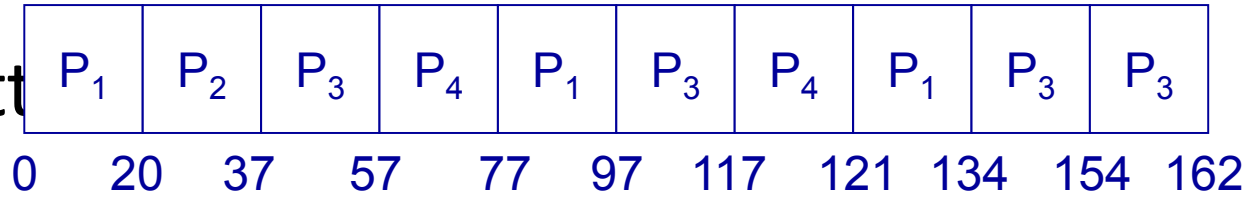
- Každý proces dostává CPU na malou jednotku času – časové kvantum
  - Desítky až stovky ms
- Po uplynutí této doby je běžící proces předběhnut nejstarším procesem ve frontě připravených procesů a zařazuje se na konec této fronty
- Je-li ve frontě připravených procesů n procesů a časové kvantum je *q*, pak každý proces získává *1/n* doby CPU, najednou nejvýše *q*

Round Robin (RR) (2)

- Žádný proces nečeká na přidělení CPU déle než *(n-1)q* časových jednotek
- Výkonnostní hodnocení
  - *q* velké → ekvivalent FCFS
  - *q* malé → velká režie; v praxi musí být *q* musí být dostatečně velké s ohledem na režii přepínání kontextu
  - Zlaté pravidlo volby *q* – 80 % dávek CPU by mělo být < *q*

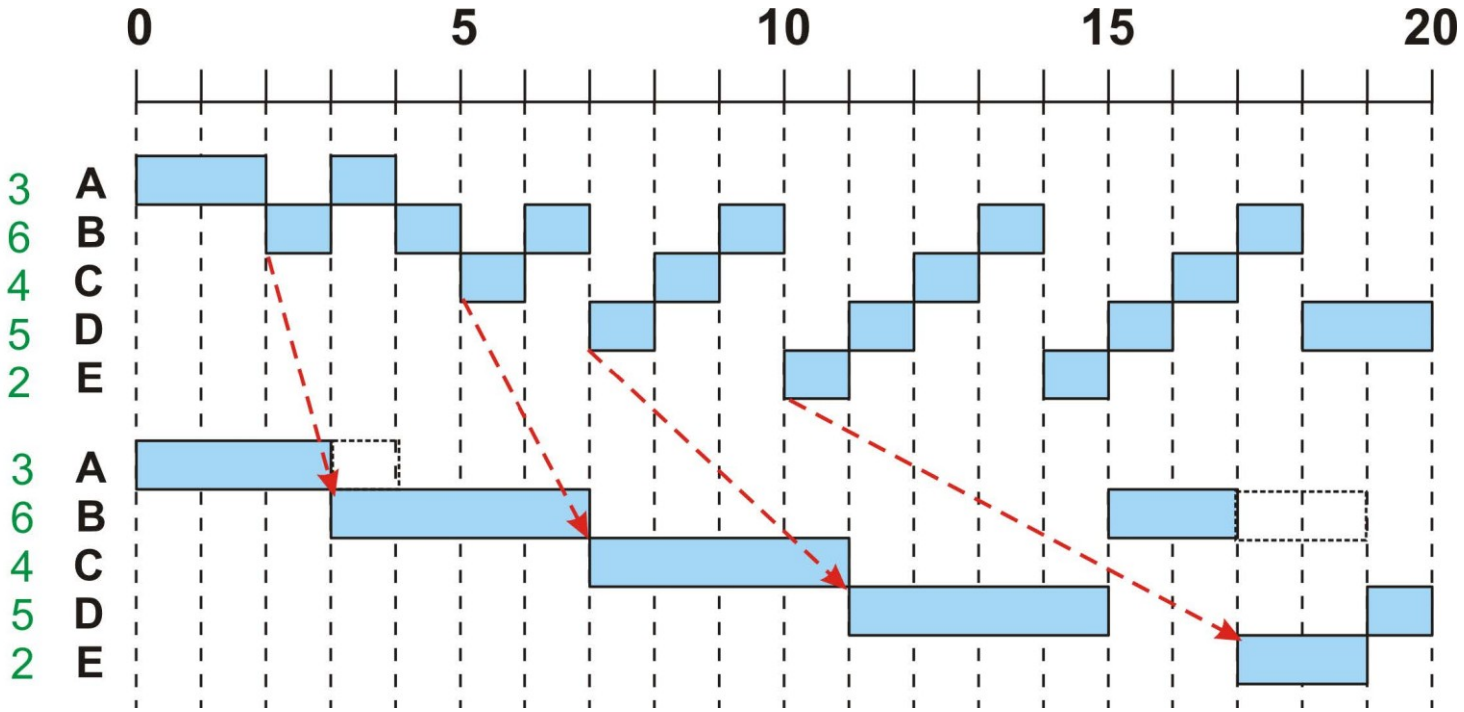
RR s časovým kvantem = 20

- Proces Délka dávky CPU
  - P1 53
  - P2 17
  - P3 68
  - P4 24
- Gantt



- Typicky se dosahuje delší průměrné doby

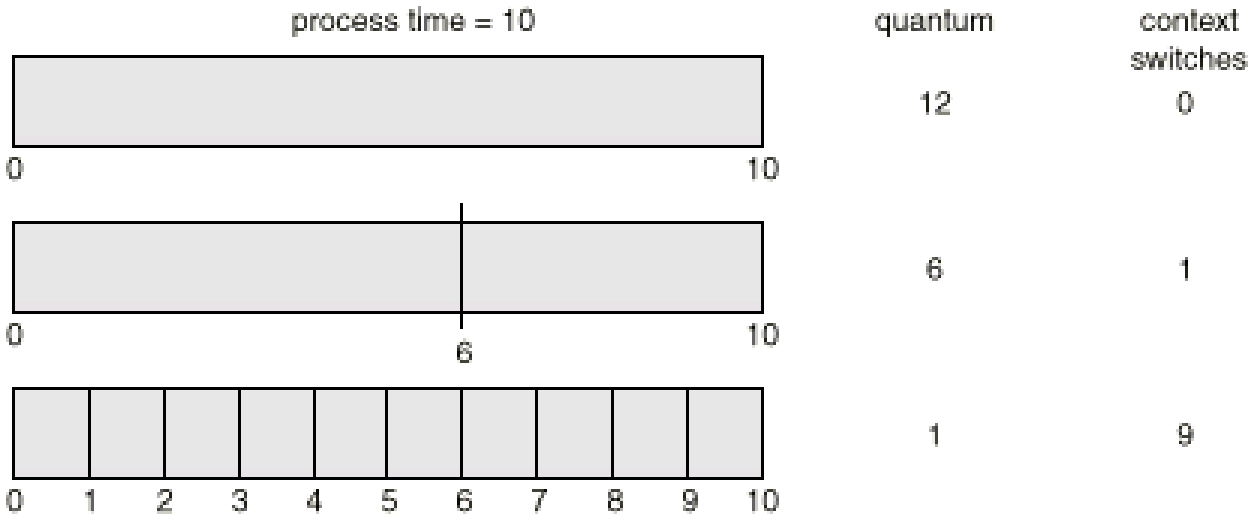
Proces	Čas příchodu	Doba realizce
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2



Round-Robin (RR), $q = 1$

Round-Robin (RR), $q = 4$

Časové kvantum a doba přepnutí kontextu



- Příklad
  - Doba přepnutí kontextu = 0,01
  - Ztráty související s režií OS při q = 12, 6 a 1 jsou 0,08 %; 0,16 % a 1 %

Algoritmus SJF (1)

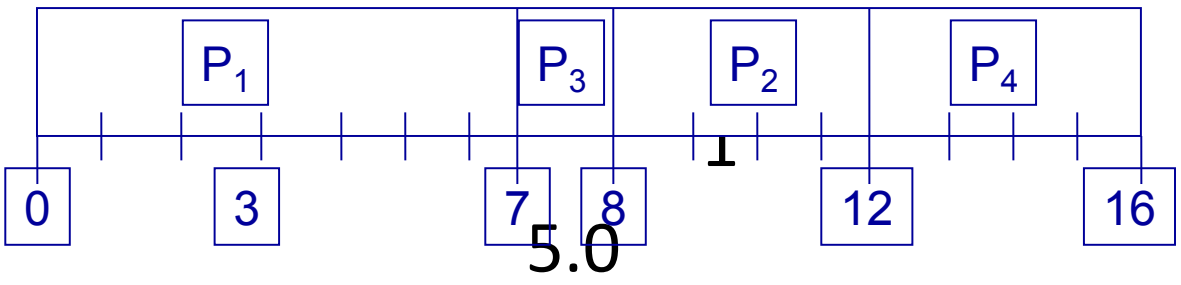
- Algoritmus Shortest-Job-First (Shotest-Process-Next)
- Musíme znát délku příštího požadavku na dávku CPU pro každý proces
- Vybírá se proces s nejkratším požadavkem na CPU
- SJF je optimální algoritmus (pro danou množinu procesů dává minimální **průměrnou** dobu čekání)

Algoritmus SJF (1)

- Dvě varianty
  - Nepreemptivní, bez předbíhání
    - Jakmile se CPU předá vybranému procesu, tento nemůže být předběhnut žádným jiným procesem, dokud přidělenou dávku CPU nedokončí
  - Preemptivní, s předbíháním
    - Jakmile se ve frontě připravených procesů objeví proces s délkou dávky CPU kratší než je doba zbývající k dokončení dávky právě běžícího procesu, je právě běžící proces ve využívání CPU předběhnut novým procesem
    - Tato varianta se rovněž nazývá Shortest-Remaining-Time-First (SRTF)

Nepreemptivní algoritmus SJF

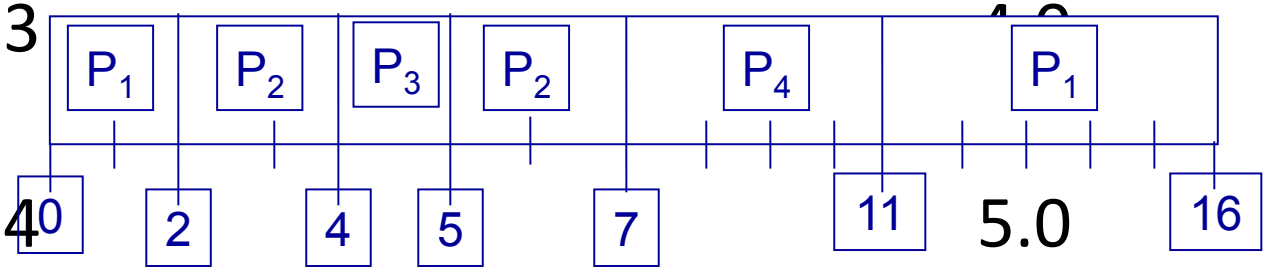
Proces	Doba příchodu	Délka dávky CPU
– P1	0.0	7
– P2	2.0	4
– P3	4.0	1
– P4	5.0	4



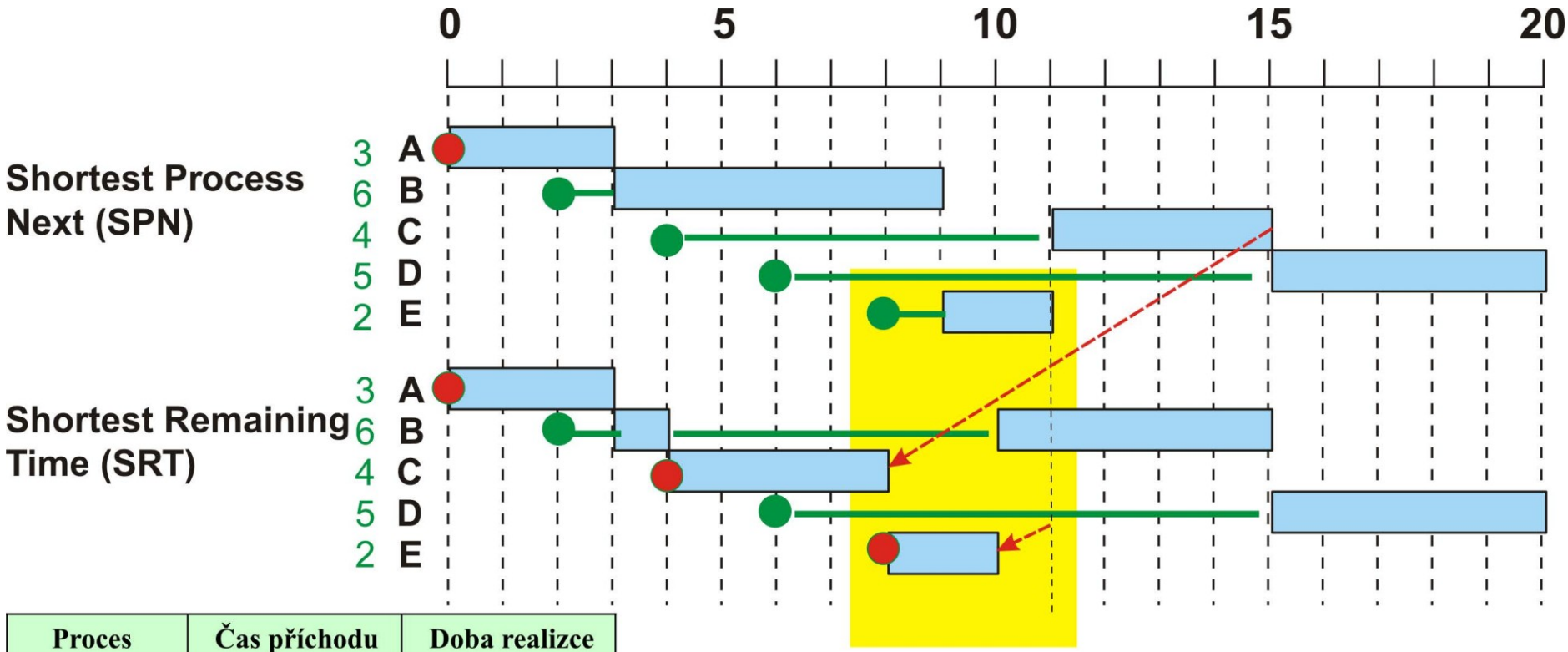
- Ganttovo schématické vyjádření plánu

Preemptivní algoritmum SJF

- Proces Doba příchodu Délka dávky CPU
  - P1 0.0 7
  - P2 2.0 4
  - P3
  - P4 5.0 4



- Ganttovo schématické vyjádření plánu



Proces	Čas příchodu	Doba realizce
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2

Prioritní plánování (1)

- S každým procesem je spojeno prioritní číslo
  - Prioritní číslo – preference procesu pro výběr příště běžícího procesu
  - CPU se přiděluje procesu s největší prioritou
  - Nejvyšší prioritě obvykle odpovídá nejnižší prioritní číslo ;-)
- Opět dvě varianty
  - Nepreemptivní, bez předbíhání
    - Jakmile proces získá přístup k CPU nemůže být předběhnut jiným procesem dokud dávku neukončí
  - Preemptivní, s předbíháním

Prioritní plánování (2)

- SJF je prioritní plánování, prioritou je předpokládaná délka příští CPU dávky
- Stárnutí
  - Procesy s nižší prioritou se nemusí nikdy provést
  - Řešení
    - Zrání – priorita se s postupem času zvyšuje

Příklad – Win32 (1)

- Procesy při vytvoření přiděleny do jedné z následujících tříd
  - Idle
  - Below Normal
  - Normal
  - Above Normal
  - High
  - Realtime

Příklad – Win32 (2)

- Plánovací algoritmus je řízen především prioritami
  - 32 front (FIFO seznamů) vláken, která jsou „připravena“
    - Pro každou úroveň priority jedna fronta
    - Fronty jsou společné pro všechny procesory
  - Když je vlákno „připraveno“
    - Buď běží okamžitě
    - Nebo je umístěno na konec fronty „připravených“ procesů ve své prioritě
  - Na jednoprocesorovém stroji vždy běží vlákno s nejvyšší prioritou

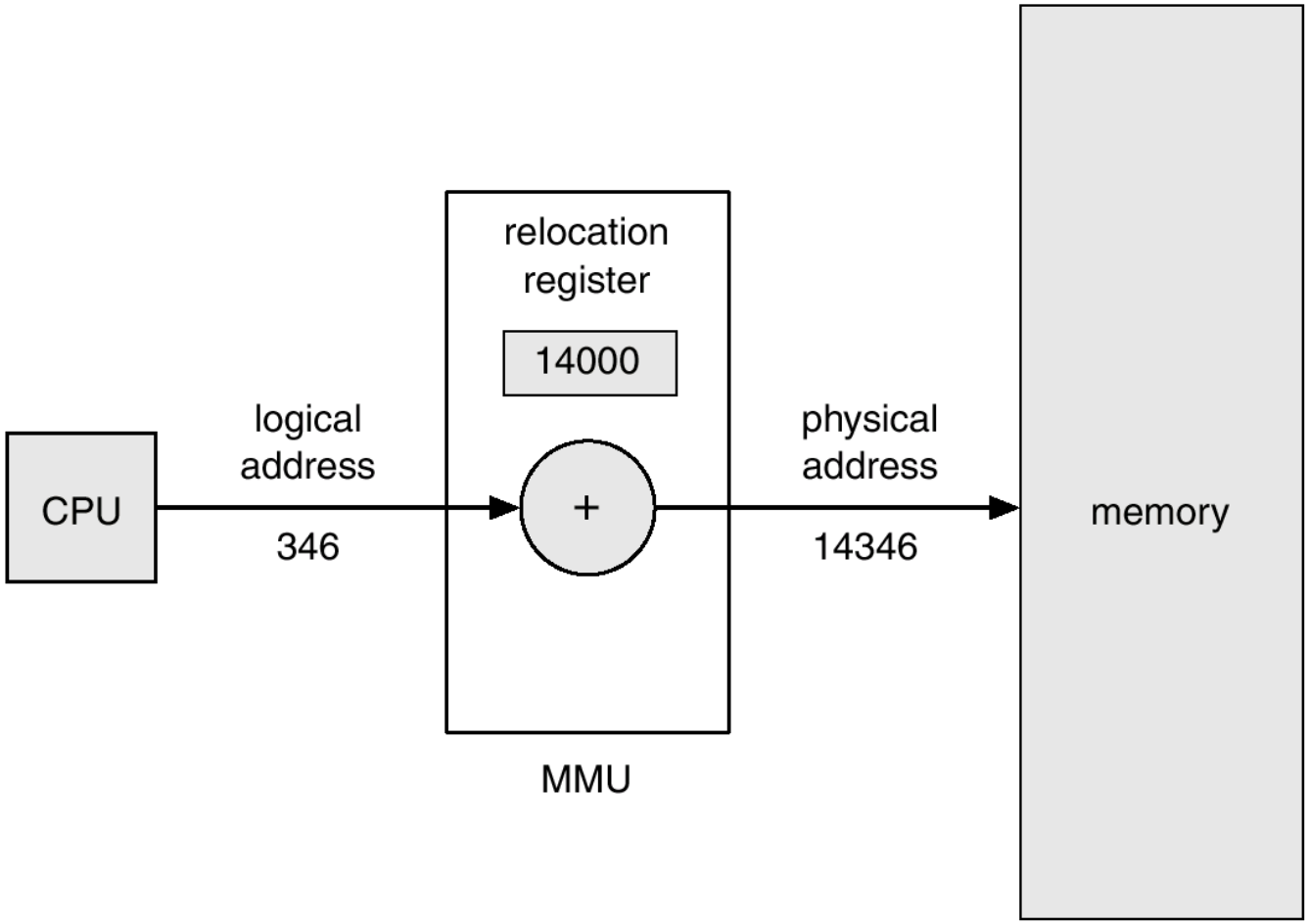
Paměť – principy, základy

- Pro běh procesu je nutné, aby program, který je vykonáván byl umístěn v operační paměti (hlavní paměti)
  - Včetně dat
- Z programu se stává proces (aktivní entita schopná spuštění na CPU) provedením celé řady kroků
  - Naplnění tabulek, umístění do operační paměti
  - Vázání adres instrukcí a dat na adresy operační paměti

Memory-Management Unit

- Hardwarový modul převádějící logické adresy na fyzické adresy
- Uživatelský program pracuje s logickými adresami, uživatelský program nevidí fyzické adresy
- Připočítává se obsah „relokačního registru“ k adresám generovaným uživatelským procesem v okamžiku, kdy je předávána jako ukazatel do operační paměti

Relokační registr



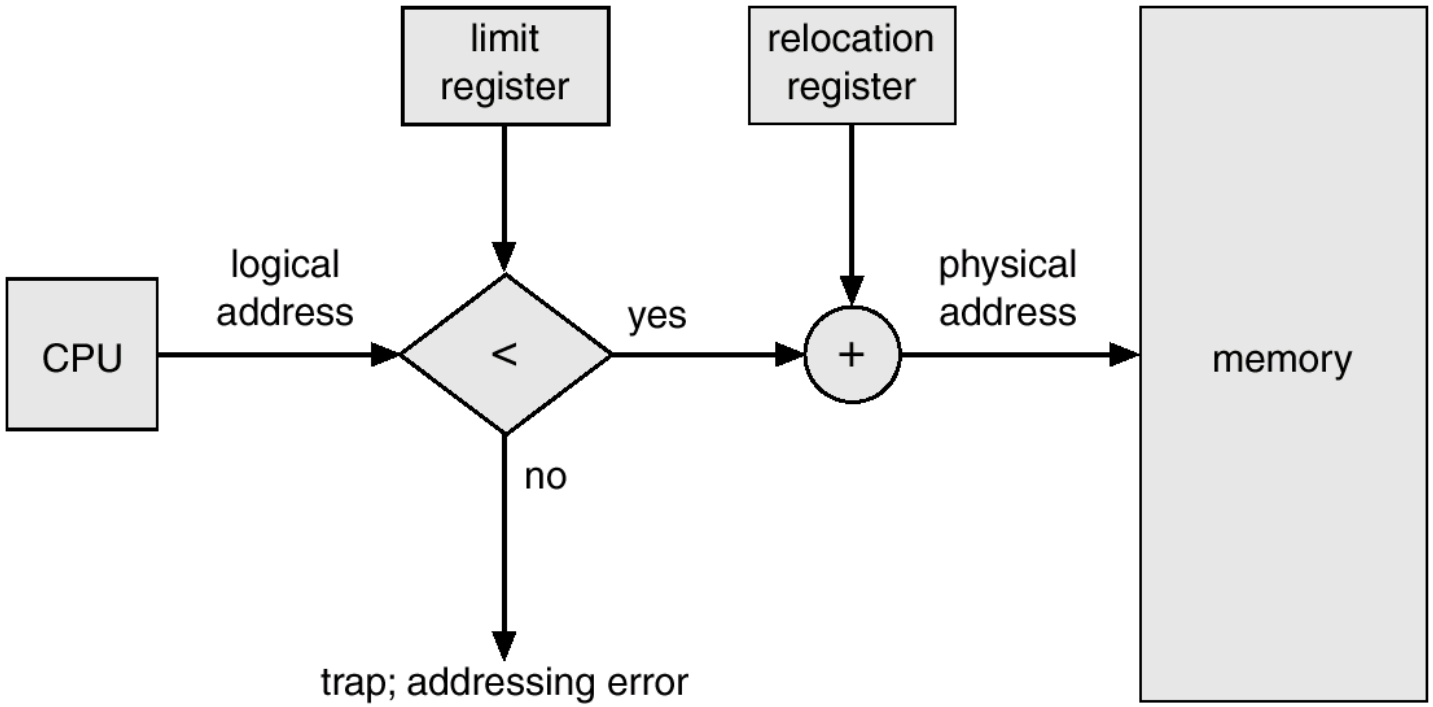
Adresový prostor

- Logický adresový prostor (LAP), fyzický adresový prostor (FAP)
  - LAP – (logická adresa, virtuální adresa) dána adresou ve strojovém jazyku, generuje CPU
  - FAP – (fyzická) adresa akceptovaná operační pamětí
- Logické a fyzické adresové prostory se shodují v době kompilace a v době zavádění
- Logické a fyzické adresové prostory mohou být rozdílné při vázání v době běhu

Souvislé oblasti

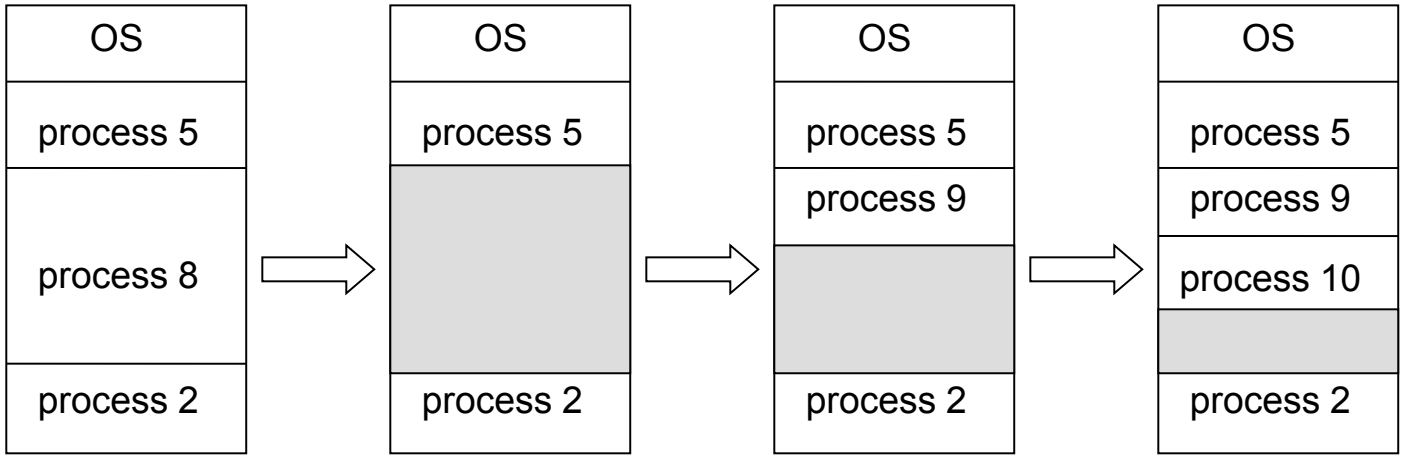
- Operační paměť se dělí do dvou sekcí
  - Rezidentní OS, obvykle na počátku FAP s tabulkou ovladačů přerušení
  - Uživatelské procesy
- Přidělování jedné souvislé části paměti
  - Pro ochranu procesů uživatelů mezi sebou a OS lze použít schéma s relokačním registrem
  - Relokační registr – hodnota nejmenší fyzické adresy paměti procesu
  - Mezní registr – rozpětí logických adres, logická adresa musí být menší nebo rovna mezním

HW podpora


limit register
relocation register
logical address
physical address
CPU
<
yes
+
memory
no
trap; addressing error

Souvislé oblasti

- Přidělování několika částí paměti
  - *Díra* – blok dostupné paměti
    - Bloky jsou roztroušeny po FAP
  - Evidenci o přidělených a volných sekcí udržuje OS



Přidělování paměti

- Kterou oblast délky n přidělit, když volná paměť je rozmístěna ve více souvislých nesousedních sekcích?
  - First-fit
    - Přiděluje se první dostatečně dlouhá volná oblast resp. její počátek
  - Best-fit
    - Přiděluje se nejmenší dostatečně dlouhá volná oblast resp. její počátek
    - Generují se velmi malé (nejmenší) možné volné díry
  - Worst-fit

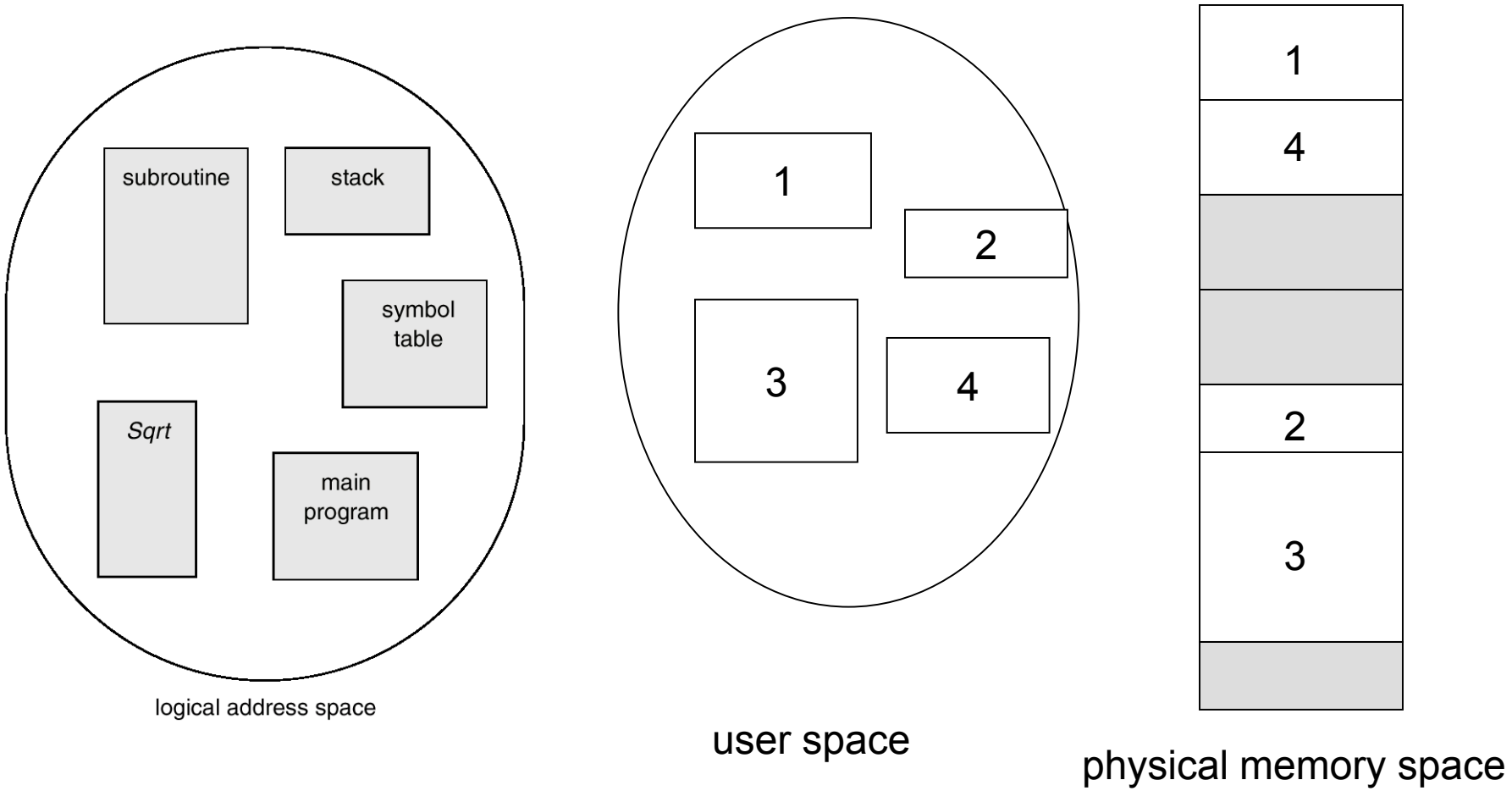
Problém fragmentace

- Vnější fragmentace
  - Souhrn volné paměti je dostatečný, ale ne v dostatečné souvislé oblasti
- Vnitřní fragmentace
  - Přidělená oblast paměti je větší než požadovaná velikost, tj. část přidělené paměti je nevyužitá
- Snižování vnější fragmentace setřásáním
  - Přesouvají se obsahy paměti s cílem vytvořit (jeden) velký volný blok
  - Použitelné jen když je možná dynamická relokace (viz MMU)

Stránkování

- LAP procesu nemusí být jedinou souvislou sekcí FAP, LAP se zobrazuje do (po částech volných) sekcí FAP
- FAP se dělí na sekce zvané rámce (*frames*)
  - Pevná délka, délka v násobcích mocnin 2 (obvykle mezi 512 až 8192 bajty)
- LAP se dělí na sekce zvané stránky (*pages*)
  - Pevná délka, shodná s délkou rámců
- Udržujeme seznam volných rámců
- Program délky *n* stránek se umístí (zavede) do

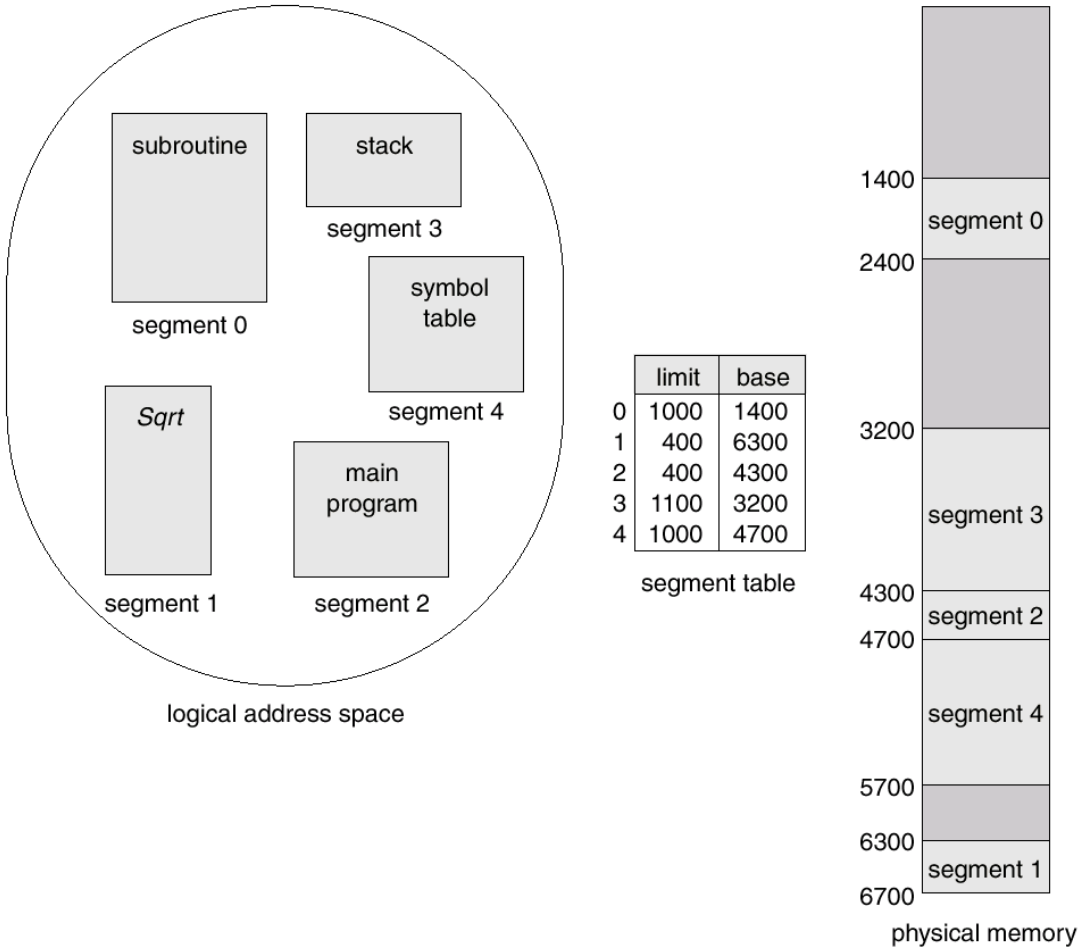
Segmentování

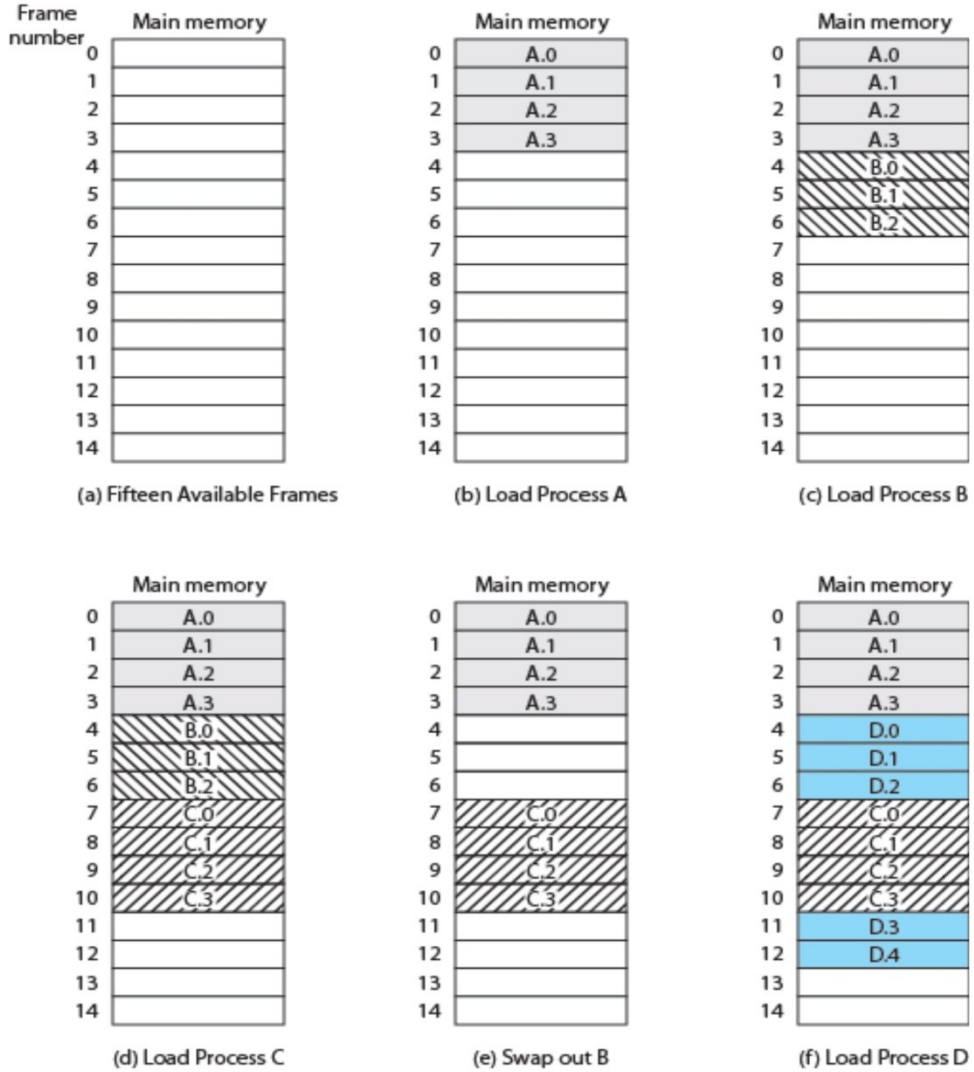


Segmentování

- Logická adresa je dvojice (segment s, offset d)
- Tabulka segmentů, Segment table, ST
  - Base – počáteční adresa umístění segmentu ve FAP
  - Limit – délka segmentu
- Segment-table base register (STBR)
  - Odkaz na umístění ST v paměti
- Segment-table length register (STLR)
  - Počet segmentů, s je legální když s < STLR
- Relokace – dynamická, pomocí ST

Příklad segmentace





0	0
1	1
2	2
3	3

Process A page table

0	—
1	—
2	—

Process B page table

0	7
1	8
2	9
3	10

Process C page table

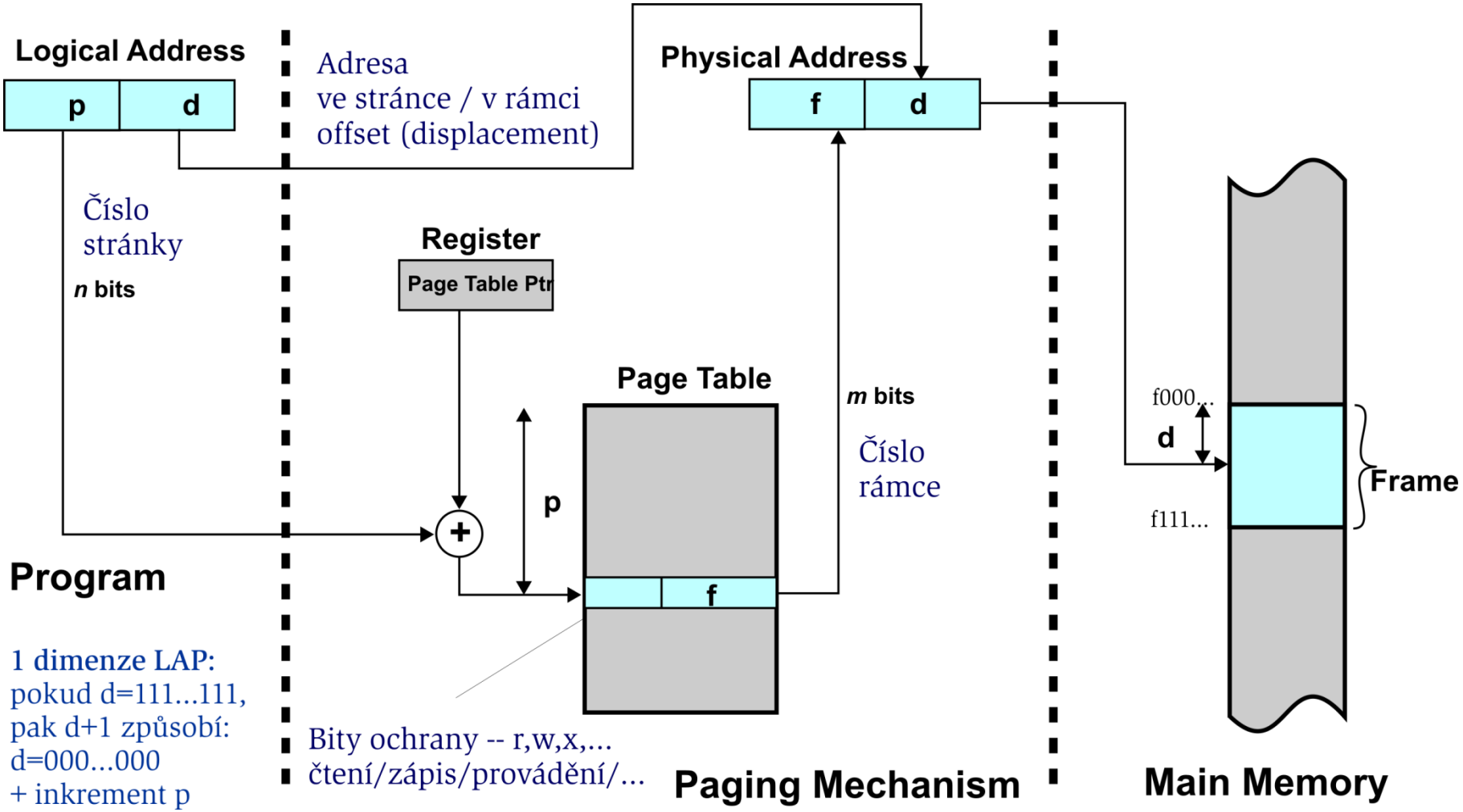
0	4
1	5
2	6
3	11
4	12

Process D page table

13
14

Free frame list

Stránkování, LAP na FAP



Stránkování a segmentování (Intel 386)