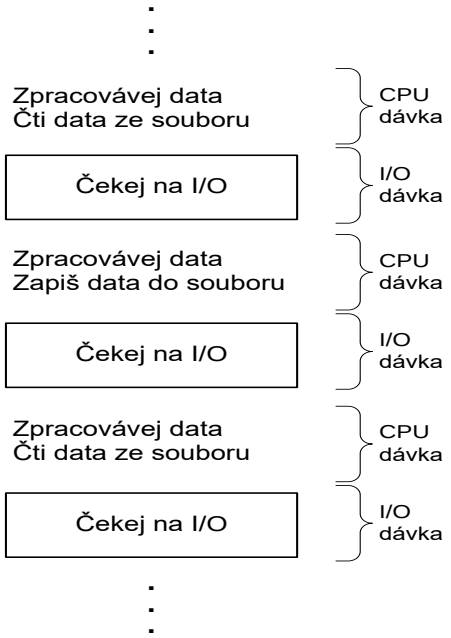
OSTRAVSKÁ UNIVERZITA
PŘÍRODOVĚDECKÁ FAKULTA

Plánování procesů, meziprocesorová komunikace

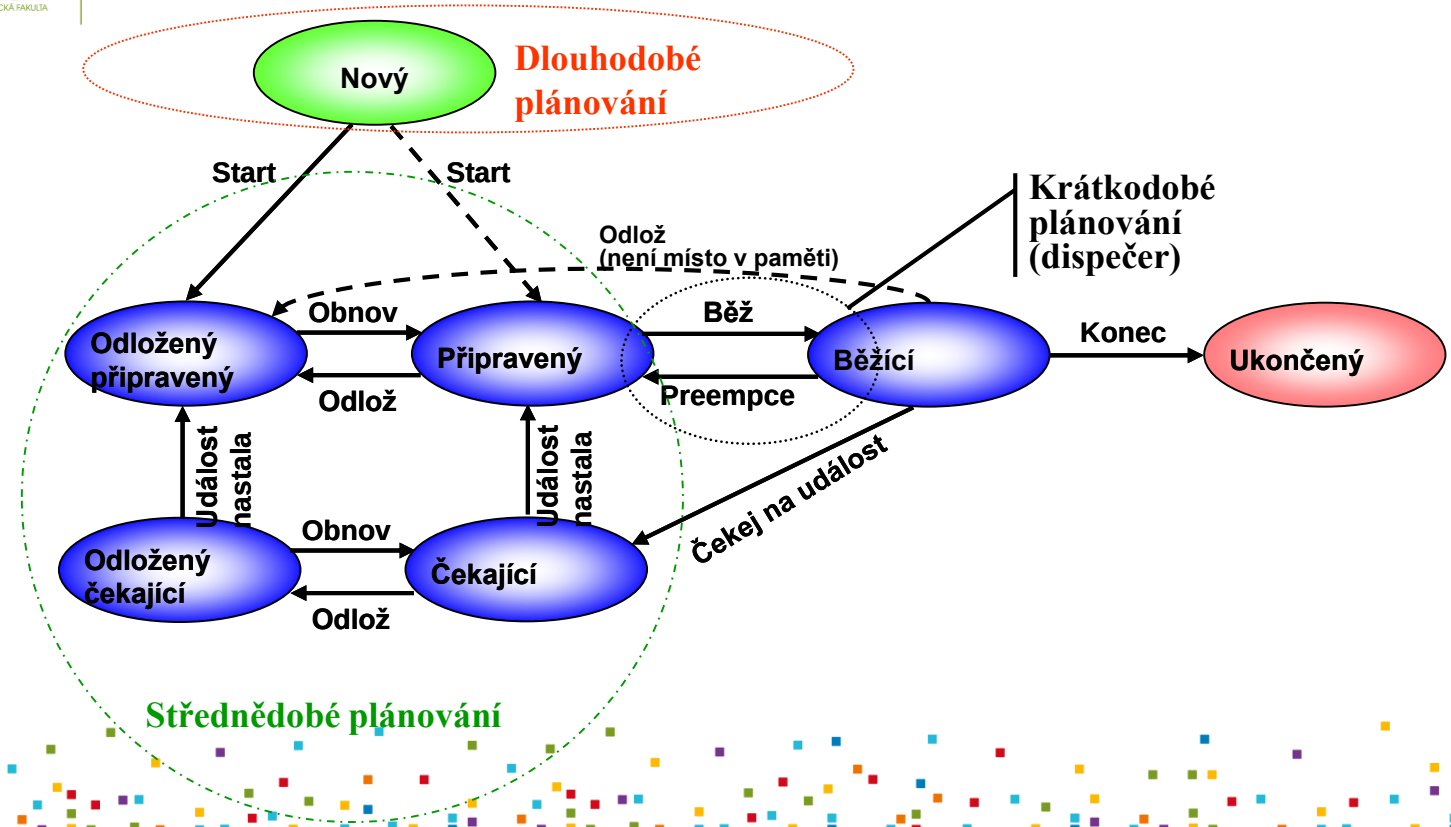
Ing. Pavel Smolka, Ph.D.

Motivace plánování CPU

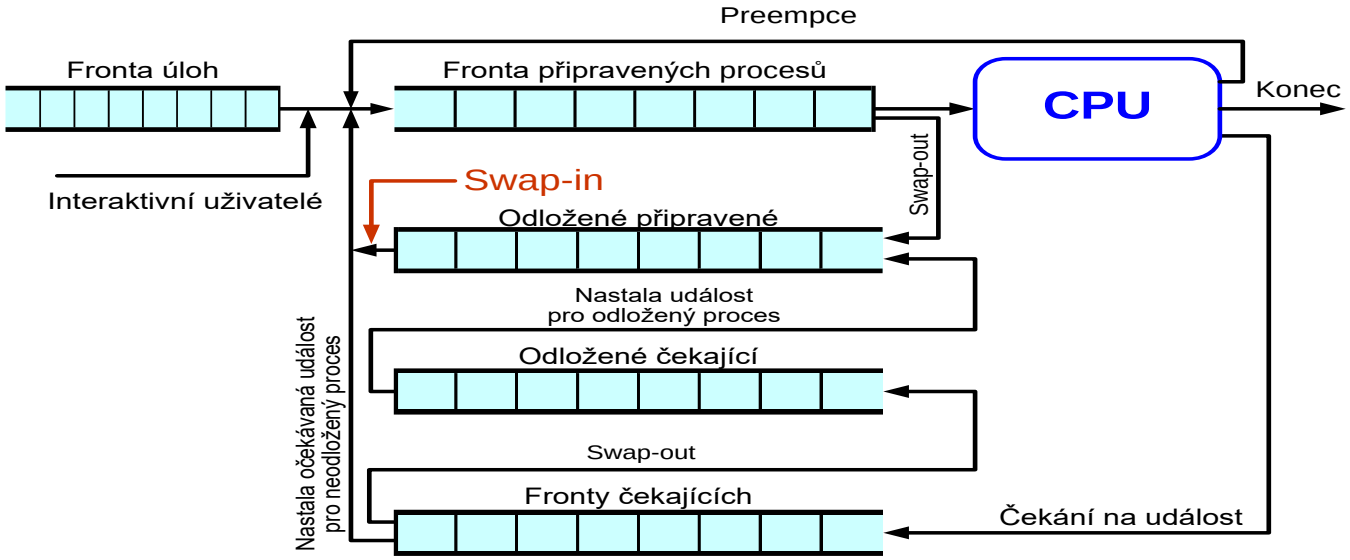
- Maximálního využití CPU se dosáhne uplatněním multiprogramování
- Proč ?
- Běh procesu = cykly alternujících dávek → [: CPU dávka, I/O dávka :]
- CPU dávka se může v čase překrývat s I/O dávkami dalších procesů



Plánovače procesů



Frontový model plánování CPU



Kritéria krátkodobého plánování

- Uživatelsky orientovaná
  - čas odezvy - doba od vzniku požadavku do reakce na něj
  - doba obrátky - doba od vzniku procesu do jeho dokončení
  - konečná lhůta (*deadline*) - požadavek dodržení stanoveného času dokončení
  - předvídatelnost
    - Úloha by měla být dokončena za zhruba stejnou dobu bez ohledu na celkovou zátěž systému
    - Je-li systém vytížen, prodloužení odezvy by mělo být rovnoměrně rozděleno mezi procesy
- Systémově orientovaná
  - Průchodnost - počet procesů dokončených za jednotku času
  - využití procesoru - relativní čas procesoru věnovaný aplikačním procesům
  - Spravedlivost - každý proces by měl dostat svůj čas (ne „hladovění“ či „stárnutí“)
  - vyvažování zátěže systémových prostředků - systémové prostředky (periferie, hlavní paměť) by měly být zatěžovány v čase rovnoměrně

Plánovač procesů

- Aktivace plánovače
  - Obslužná rutina přerušení na svém konci může ohlásit tzv. významnou událost v systému
    - např. dokončení přenosu dat, vyčerpání časového kvanta
  - Významná událost aktivuje plánovač, který rozhodne, co dále
  - Plánovač může přepnout kontext ⇒ přechod od jednoho procesu k jinému je VŽDY důsledkem nějakého PŘERUŠENÍ
- Fronta připravených procesů
  - Plánovač rozhoduje, který proces aktivovat
  - Proces v čele fronty dostává procesor a může tak způsobit preempci
  - Preempce může nastat kdykoliv (i bez „vědomí“ běžícího procesu)
- Určení priority procesů
  - Klíč k dosažení cílů plánovače (spravedlivost, propustnost, ...)
  - Odhaduje měnící se charakteristiky procesu
  - Založen na měření chování procesu v nedávné historii

Plánovací algoritmy

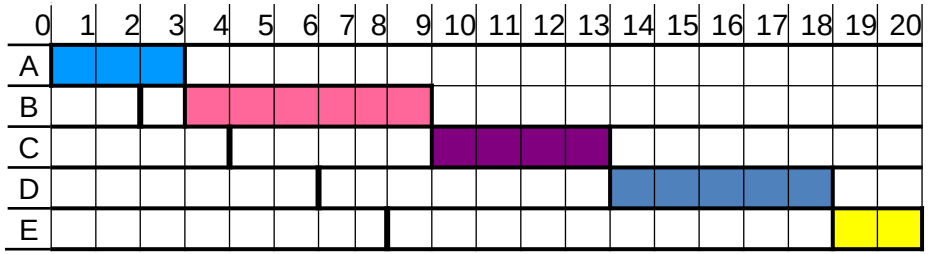
Plánovací algoritmy:

- FCFS (*First Come First Served*)
- SPN (SJF) (*Shortest Process Next*)
- SRT (*Shortest Remaining Time*)
- Cyklické (*Round-Robin*)
- Zpětnovazební (*Feedback*)

Proces	Příchod	Potřebný čas
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2

Plánování FCFS

- FCFS = *First Come First Served* – prostá fronta FIFO
- Nejjednodušší nepreemptivní plánování
- Nový proces se zařadí na konec fronty
- Průměrné čekání může být velmi dlouhé
- Příklad:

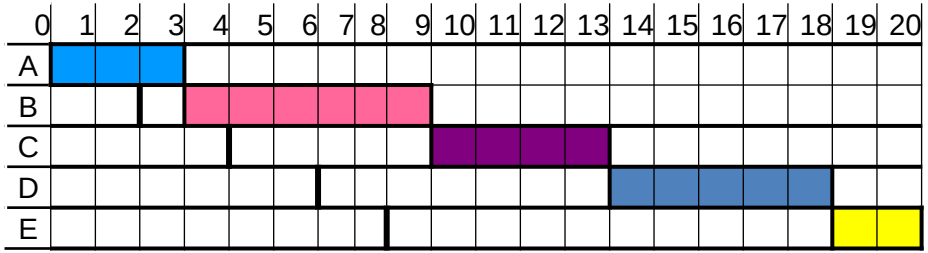


Proces	Příchod	Potřebný čas
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2

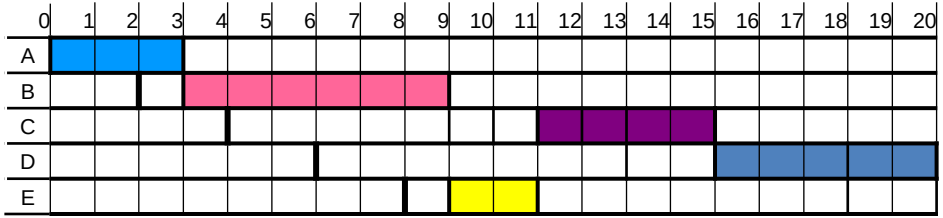
$$ {}^{w}T_{\mathrm{Avg}} = \frac{0+1+5+7+10}{5} = 4{,}6 $$

Průměrné čekání bychom mohli zredukovat:

- Např. v čase 9 je procesor volný a máme na výběr procesy C, D a E
- Spustíme-li je v pořadí podle jejich neklesajících délek, tj. E, C, D, bude průměrné čekání výrazně menší



$$^{w}T_{\text{Avg}} = \frac{0+1+5+7+10}{5} = 4{,}6$$



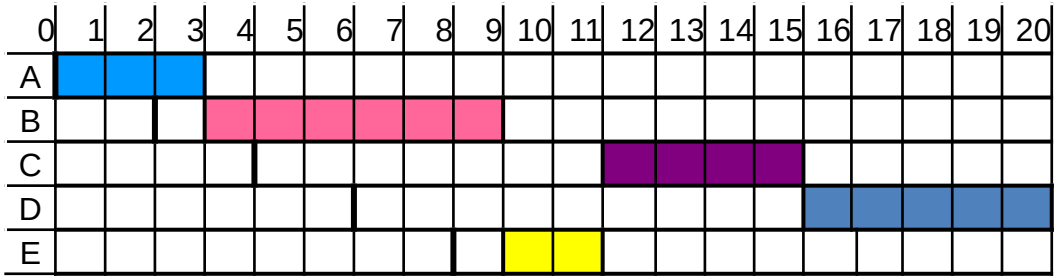
$$^{w}T_{\text{Avg}} = \frac{0+1+7+9+1}{5} = 3{,}6$$

Vlastnosti FCFS

- FCFS je primitivní nepreemptivní plánovací postup
- Průměrná doba čekání ${}^{w}T_{\text{Avg}}$ silně závisí na pořadí přicházejících dávek
- Krátké procesy následující po dlouhém procesu vytváří tzv. konvojový efekt
  - Všechny procesy čekají, až skončí dlouhý proces
- Pro krátkodobé plánování se FCFS prakticky nepoužívá.
  - Používá se pouze jako složka složitějších plánovacích postupů

Plánování SPN (SJF)

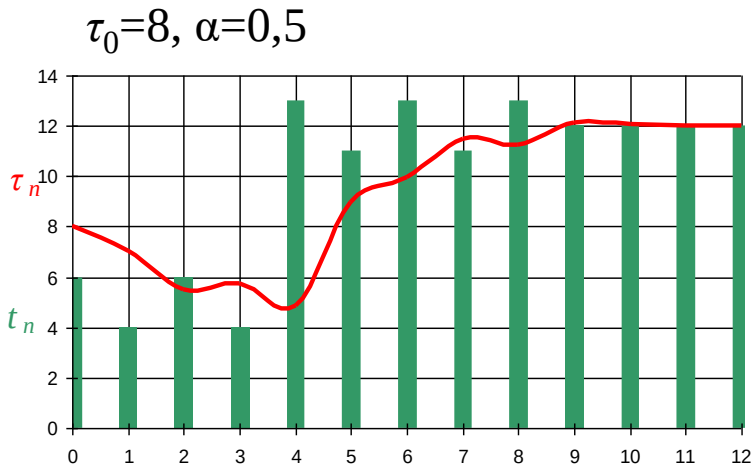
- SPN = *Shortest Process Next* (nejkratší proces jako příští); též nazýváno SJF = *Shortest Job First*
  - Opět nepreemptivní
  - Vybírá se připravený proces s nejkratší příští dávkou CPU
  - Problém s délkou příští dávky CPU
  - Krátké procesy předbíhají delší, nebezpečí stárnutí dlouhých procesů
  - Je-li kritériem kvality plánování průměrná doba čekání, je SJF optimální algoritmus, což se dá exaktně dokázat
- Příklad:



$$^{w}T_{\text{Avg}} = \frac{0+1+7+9+1}{5} = 3{,}6$$

Jak určit délku příští dávky CPU procesu

- Délka příští dávky CPU skutečného procesu je známa jen ve velmi speciálních případech
  - Proto je nutno délku dávky odhadovat na základě nedávné historie procesu
  - Nejčastěji se používá tzv. exponenciální průměrování
- Exponenciální průměrování
  - t_n … skutečná délka n-té dávky CPU
  - τ_{n+1} … odhad délky příští dávky CPU
  - α, $0 \leq \alpha \leq 1$ … parametr vlivu historie
  - $\tau_{n+1} = \alpha t_n + (1 - \alpha)\tau_n$



Příklad:

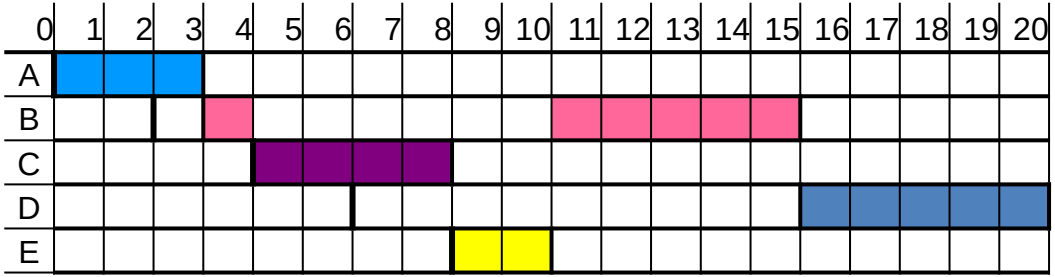
$\alpha = 0{,}5$

$\tau_{n+1} = 0{,}5t_n + 0{,}5\tau_n = 0{,}5(t_n + \tau_n)$

τ_0 se volí jako průměrná délka CPU dávky v systému nebo se odvodí z typu programu

Plánování SRT

- SRT = *Shortest Remaining Time* (nejkratší zbývající čas)
  - Preemptivní varianta SPN (SJF)
  - CPU dostane proces potřebující nejméně času do svého ukončení
  - Dojde k preempci, jestliže existuje proces, kterému zbývá k jeho dokončení čas kratší, než je čas zbývající do skončení procesu běžícího
  - Opět problém s odhadem budoucí délky dávky CPU
- Příklad:



$$^{w}T_{\text{Avg}} = \frac{0+1+0+9+0}{5} = 2{,}0$$

Prioritní plánování

- Každému procesu je přiřazeno prioritní číslo (integer)
  - Prioritní číslo – preference procesu při výběru procesu, kterému má být přiřazena CPU
  - CPU se přiděluje procesu s nejvyšší prioritou
  - Nejvyšší prioritě obvykle odpovídá nejnižší prioritní číslo
    - Ve Windows je to obráceně
- Existují se opět dvě varianty:
  - Nepreemptivní
    - Jakmile se vybranému procesu procesor předá, procesor mu nebude odňat, dokud se jeho CPU dávka nedokončí
  - Preemptivní
    - Jakmile se ve frontě připravených objeví proces s prioritou vyšší, než je priorita právě běžícího procesu, nový proces předběhne právě běžící proces a odejme mu procesor
- SPN i SRT jsou příklady prioritního plánování
  - Prioritou je predikovaná délka příští CPU dávky
  - SPN je nepreemptivní prioritní plánování
  - SRT je preemptivní prioritní plánování

OSTRAVSKÁ UNIVERZITA
PŘÍRODOVĚDECKÁ FAKULTA

Prioritní plánování

- Problém stárnutí (*starvation*):
  - Procesy s nízkou prioritou nikdy nepoběží → nikdy na ně nepřijde řada
    - Údajně: Když po řadě let vypínali v roce 1973 na M.I.T. svůj IBM 7094 (jeden z největších strojů své doby), našli proces s nízkou prioritou, který čekal od roku 1967.
- Řešení problému stárnutí: zrání procesů (*aging*)
  - Je nutno dovolit, aby se procesu zvyšovala priorita na základě jeho historie a doby setrvávání ve frontě připravených
    - Během čekání na procesor se priorita procesu zvyšuje

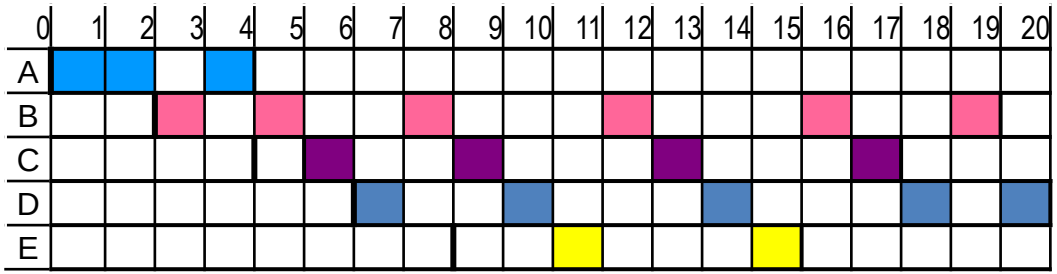
Cyklické plánování

- Cyklická obsluha (*Round-robin*) – RR
- Z principu preemptivní plánování
- Každý proces dostává CPU periodicky na malý časový úsek, tzv. časové kvantum, délky q (= desítky ms)
  - Shodná priorita všech procesů
  - Po vyčerpání kvanta je běžícímu procesu odňato CPU ve prospěch nejstaršího procesu ve frontě připravených a dosud běžící proces se zařazuje na konec této fronty
  - Je-li ve frontě připravených procesů n procesů, pak každý proces získává $1/n$-tinu doby CPU
  - Žádný proces nedostane 2 kvanta za sebou (pokud není jediný připravený)
  - Žádný proces nečeká na přidělení CPU déle než $(n-1)q$ časových jednotek

Cyklické plánování

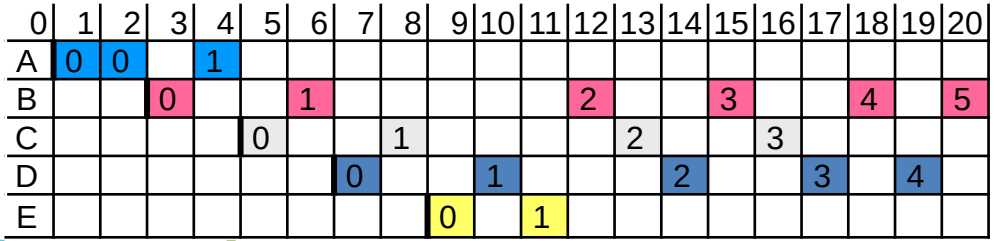
- Efektivita silně závisí na velikosti kvanta
  - Veliké kvantum – blíží se chování FCFS
    - Procesy dokončí svoji CPU dávku dříve, než jim vyprší kvantum.
  - Malé kvantum => časté přepínání kontextu => značná režie
- Typicky
  - Dosahuje se průměrné doby obrátky delší oproti plánování SRT
  - Výrazně lepší je čas odezvy
  - Průměrná doba obrátky se může zlepšit, pokud většina procesů se době *q* ukončí
  - Empirické pravidlo pro stanovení *q*: cca 80% procesů by nemělo vyčerpat kvantum

Příklad:

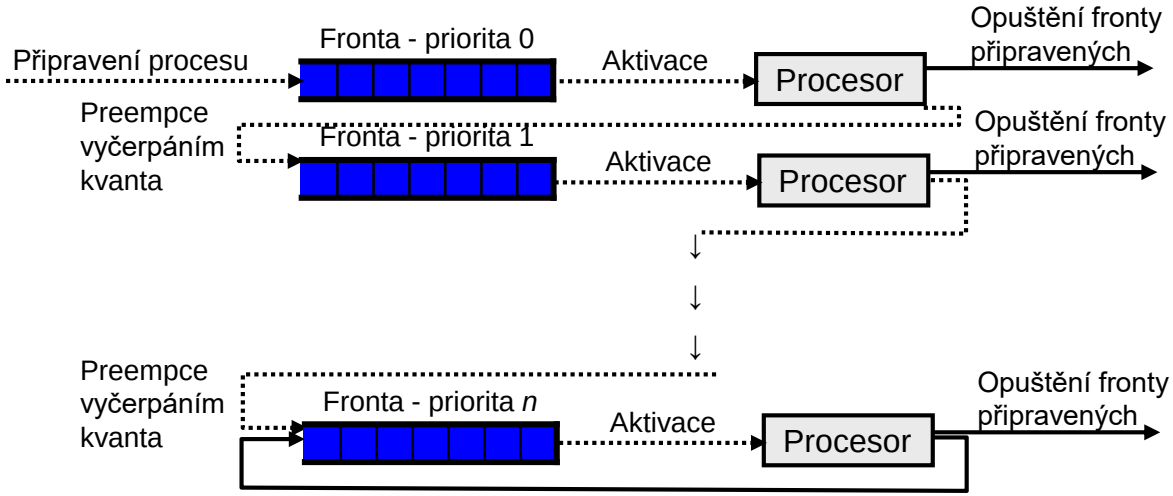


Zpětnovazební plánování

- Základní problém:
  - Neznáme předem časy, které budou procesy potřebovat
- Východisko:
  - Penalizace procesů, které běžely dlouho
- Řešení:
  - Dojde-li k preempci přečerpáním časového kvanta, procesu se snižuje priorita
  - Implementace pomocí víceúrovňových front
    - pro každou prioritu jedna
  - Nad každou frontou samostatně běží algoritmus určitého typu plánování
    - obvykle RR s různými kvanty a FCFS pro frontu s nejnižší prioritou
- Příklad

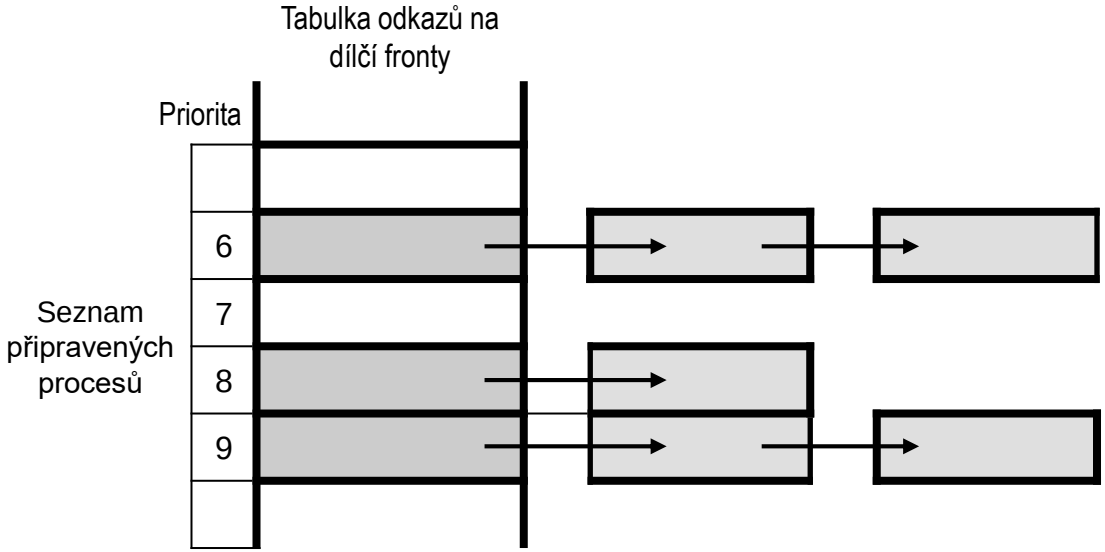


Víceúrovňové fronty



- Proces opouštějící procesor kvůli vyčerpání časového kvanta je přeřazen do fronty s nižší prioritou
- Fronty s nižší prioritou mohou mít delší kvanta
- Problém stárnutí ve frontě s nejnižší prioritou
  - Řeší se pomocí zrání (*aging*) – v jistých časových intervalech se zvyšuje procesům priorita, a tak se přemísťují do „vyšších“ front

Implementace víceúrovňových front



- Implementace JOS musí dbát na rychlost přístupu k datovým strukturám, aby přepínání kontextu bylo co nejrychlejší
- Fronta na procesor je rozdělena na dílčí fronty, pro každou prioritu jedna samostatně uspořádaná způsobem FIFO.

Obecný plánovač

OSTRAVSKÁ UNIVERZITA
PŘÍRODOVĚDECKÁ FAKULTA

Tři komponenty:

1. Rozhodování - kterému procesu přidělit procesor (resp. i který z více procesorů)
2. Prioritní funkce - všem připraveným procesům určit efektivní priority
3. Arbitrážní pravidlo - co činit, jsou-li dva procesy rovnocenné (se shodnou prioritou)

Rozhodování

- Pracuje nad frontou připravených
- Rozhoduje se když:
  1. Nový proces se stane připraveným
  2. Běžící proces skončí
  3. Čekající proces změní svůj stav na připravený
  4. Běžící proces se začne čekat
  5. Běžící proces vyčerpá časové kvantum
  6. Připravenému procesu vzroste priorita nad prioritu procesu běžícího

Obecný plánovač

OSTRAVSKÁ UNIVERZITA
PŘÍRODOVĚDECKÁ FAKULTA

Prioritní funkce

- Určuje efektivní prioritu připravených procesů
- Může záviset na vlastnostech procesů
- Základní prioritní úroveň
  - Vysoká pro interaktivní procesy
  - Nízká pro dávkové zpracování „na pozadí“
- Nároky na paměť (velká režie odkládání):
  - „Malý“ proces – rychlé odkládání, lze snáze obsluhovat mnoho malých procesů
- Časové vlastnosti procesu
  - Relativní spotřeba časových kvant
  - Celkový spotřebovaný čas
  - Vyšší priorita krátkých procesů

Arbitrážní pravidlo

- Aplikuje se na připravené procesy se stejnou efektivní prioritou
- Náhodná volba - používá se zřídka
- Chronologické řazení - nejčastější – klasická fronta (FIFO); proces s prioritou se řadí na konec fronty

Plánování v multiprocesorech

Přiřazování procesů (vláken) procesorům:

- Architektura „master/slave“
  - Klíčové funkce jádra běží vždy na jednom konkrétním procesoru
  - Master odpovídá za plánování
  - Slave žádá o služby mastera
  - Nevýhoda: dedikace - přetížený master se stává úzkým místem systému
  - Symetrický multiprocesing (SMP)
    - Všechny procesory jsou si navzájem rovny
    - Funkce jádra mohou běžet na kterémkoliv procesoru
  - SMP vyžaduje podporu vláken v jádře

Proces musí být dělen na vlákna, aby SMP byl účinný

- Aplikace je sada vláken pracujících paralelně do společného adresního prostoru
- Vlákno běží nezávisle na ostatních vláknech procesu
- Vlákna běžící na různých procesorech dramaticky zvyšují účinnost systému

OSTRAVSKÁ UNIVERZITA
PŘÍRODOVĚDECKÁ FAKULTA

Symetrický multiprocesing (SMP)

Jedna společná (globální) fronta pro všechna vlákna

- Fronta „napájí“ společnou sadu procesorů
  - fronta může být víceúrovňová dle priorit
  - Každý procesor si sám vyhledává příští vlákno (Přesněji: instance plánovače běžící na procesoru si je sama vyhledává ...)

Fakta:

- Plánovací politiky pro přidělování procesorů v multiprocesoru nemají takový význam jako v monoprocesoru
- Možnost souběžného běhu vláken jednoho procesu na více procesorech zvyšuje potenciálně dostupný výkon pro běh aplikací

Problémy

- Jedna centrální fronta připravených sledů vyžaduje používání vzájemného vylučování v jádře
  - Kritické místo v okamžiku, kdy si hledá práci více procesorů
  - Předběhnutá (přerušená) vlákna nebudou nutně pokračovat na stejném procesoru – nelze proto plně využívat cache paměti procesorů
- Používáno ve: Windows, Linux, Mac OS X, Solaris, BSD4.4

OSTRAVSKÁ UNIVERZITA
PŘÍRODOVĚDECKÁ FAKULTA

Poznámky k plánování v multiprocesorech

- Dynamické vyvažování (*Load Balancing*) využití jednotlivých procesorů:
  - Používá se v systémech, kde každý procesor má svoji frontu připravených
  - V systémech s globální frontou není dynamické vyvažování potřebné
    - Problémem naopak je, že pokud jsou všechny sledy instrukcí (CPU dávky) v jedné společné frontě připravených, nemohou se jednoduše spustit současně a běžet paralelně
- Používají se různá (heuristická) pravidla (i při globální frontě):
  - Afinita vlákna k procesoru – použij procesor, kde vlákno již běželo (možná, že v cache procesoru budou ještě údaje z minulého běhu)
  - Použij nejméně využívaný procesor
- Mnohdy značně složité
  - při malém počtu procesorů (≤ 4) může přílišná snaha o optimalizaci plánování vést až k poklesu výkonu systému

Systémy reálného času (RT)

- Obvykle malé systémy se specializovaným použitím
  - Často vestavěné (*embedded*)
- Správná funkce systému závisí nejen na logickém (či numerickém) výsledku ale i na čase, kdy bude výsledek získán
  - Správně určený výsledek dodaný pozdě je k ničemu
- Úlohy a procesy reagují na události pocházející zvenčí systému a navenek dodávají své výsledky
  - Nastávají v „reálném čase“ a potřebná reakce musí být včasná
- Příklady
  - Řízení laboratorních či průmyslových systémů
  - Robotika
  - Řízení letového provozu
  - Telekomunikační aplikace (digitální ústředny)
  - Vojenské systémy velení a řízení
  - …

Charakteristiky OS RT

- Determinismus
  - Operace jsou prováděny ve fixovaných, předem určených časech nebo časových intervalech
  - Reakce na přerušení musí proběhnout tak, aby systém byl schopen obsluhy všech požadavků v požadovaném čase (včetně vnořených přerušení)
- Uživatelské řízení
  - Uživatel (návrhář systému) specifikuje:
    - Priority
    - Práva procesů
    - Co musí vždy zůstat v paměti
- Spolehlivost
  - Degradace chování může mít katastrofální důsledky
- Zabezpečení
  - Schopnost systému zachovat v případě chyby aspoň částečnou funkcionalitu a udržet maximální množství dat

OSTRAVSKÁ UNIVERZITA
PŘÍRODOVĚDECKÁ FAKULTA

Požadavky na OS RT

- Extrémně rychlé přepínání mezi procesy či vlákny
- OS musí být malý
- Multiprogramování s meziprocesními komunikačními nástroji
  - semafory, signály, události ➡
- Speciální souborové systémy s velkou rychlostí
  - RAM disky, souvislé soubory
- Plánování založené na prioritách
  - Pozor: preempce je ale časově náročná
- Minimalizace časových úseků, kdy je vypnut přerušovací systém
- Zvláštní hardwarové vybavení
  - hlídací časovače (*watch-dog timers*) a alarmy

Plánování CPU v RT systémech

- Tabulkou řízené statické plánování
  - Určuje pevně, kdy bude který proces spuštěn tak, aby včas skončil
  - Nejčastější případ v uzavřených systémech s předem známými procesy a jejich vlastnostmi
- Preemptivní plánování se statickými prioritami
  - Používá klasický prioritní plánovač s fixními prioritami
    - Může být problematické kvůli velké režii spojené s preempcí
- Dynamické plánování
  - Za běhu určuje proveditelnost (splnitelnost požadavků)
  - V tzv. přísných RT systémech (*Hard real-time systems*) téměř nepoužitelné vlivem velké režie
    - *Hard real-time systems* musí přísně zaručovat dokončení časově kritických procesů do předepsaného termínu

OSTRAVSKÁ UNIVERZITA
PŘÍRODOVĚDECKÁ FAKULTA

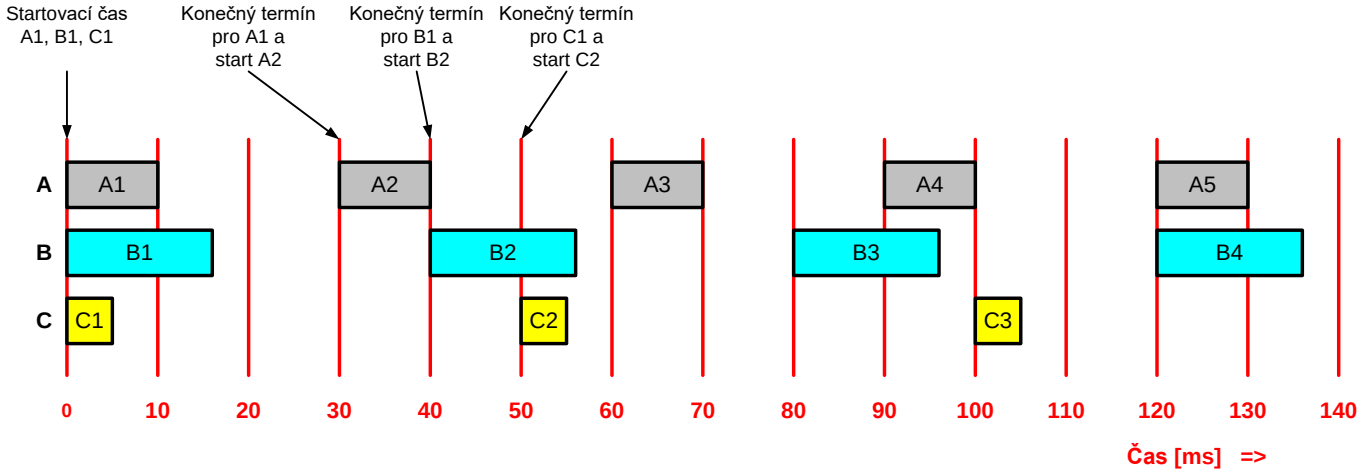
Periodické plánování dle konečného termínu

- Procesům jde zejména o včasné dokončení v rámci zadané periody běžících procesů
  - Např. v daných intervalech je třeba vzorkovat napětí z čidel
- O každém procesu je znám
  - Potřebný čas (horní hranice délky CPU dávky)
  - Termín začátku a nejzazšího konce každého běhu periodicky spouštěného procesu
- Předpoklady (zjednodušení)
  - Termín dokončení je identický s počátkem následující periody
  - Požadavky na systémové prostředky budeme ignorovat

Příklad 1

- 3 periodické procesy

Proces	Perioda p_i	Procesní čas t_i	t_i / p_i
A	30	10	0,333
B	40	15	0,375
C	50	5	0,100



Plánovatelnost v periodických úlohách

- Relativní využití strojového času
  - Proces *i* využije poměrnou část $r_i = \frac{t_i}{p_i}$ celkového strojového času, kde t_i je čas běhu a p_i je jeho perioda
  - Celkové využití je $r = \sum_{i=1}^{N} r_i = \sum_{i=1}^{N} \frac{t_i}{p_i}$
  - Aby vše mohlo pracovat musí platit (podmínka plánovatelnosti) $r = \sum_{i=1}^{N} \frac{t_i}{p_i} \leq 1$
- Náš Příklad 1 $r = \sum_{i=1}^{3} \frac{t_i}{p_i} = \frac{10}{30} + \frac{15}{40} + \frac{5}{50} = 0{,}808 < 1$

Plánování RMS

- RMS = *Rate Monotonic Scheduling*
- Statické priority
  - $Prio_i \approx p_i$ (kratší perioda = menší číslo ~ vyšší priorita)
- Použitelné pro procesy s následujícími vlastnostmi
  - Periodický proces musí skončit během své periody
  - Procesy jsou vzájemně nezávislé
  - Každý běh procesu (CPU dávka) spotřebuje konstantní čas
  - Preempce nastává okamžitě (bez režie)

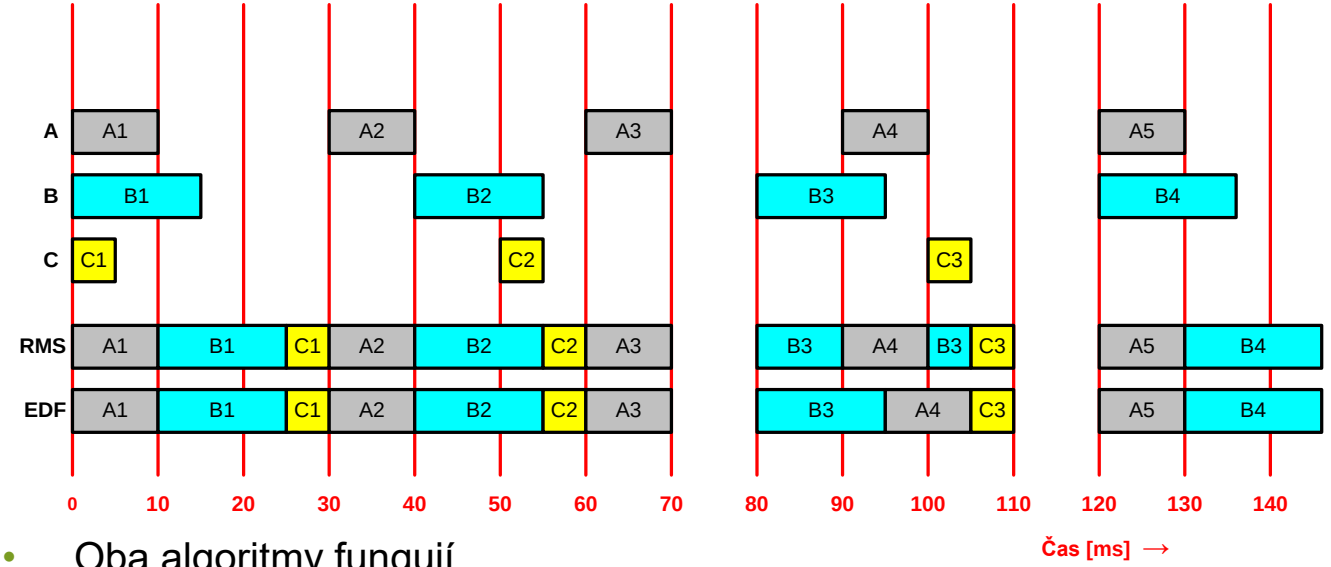
OSTRAVSKÁ UNIVERZITA
PŘÍRODOVĚDECKÁ FAKULTA

Plánování EDF

- EDF = *Earliest Deadline First*
  - Upřednostňuje proces s nejbližším termínem dokončení
- Dynamické priority
  - Plánovač vede seznam připravených procesů uspořádaný podle požadovaných časů dokončení a spustí vždy ten s nejbližším požadovaným termínem dokončení
- Vhodné pro procesy s následujícími vlastnostmi
  - Procesy nemusí být přísně periodické ani nemusí mít konstantní dobu běhu
  - Pro každý proces musí být znám požadovaný termín dokončení (*deadline*)

Příklad 1 - pokračování

Proces	Perioda p_i	Procesní čas t_i
A	30	10
B	40	15
C	50	5



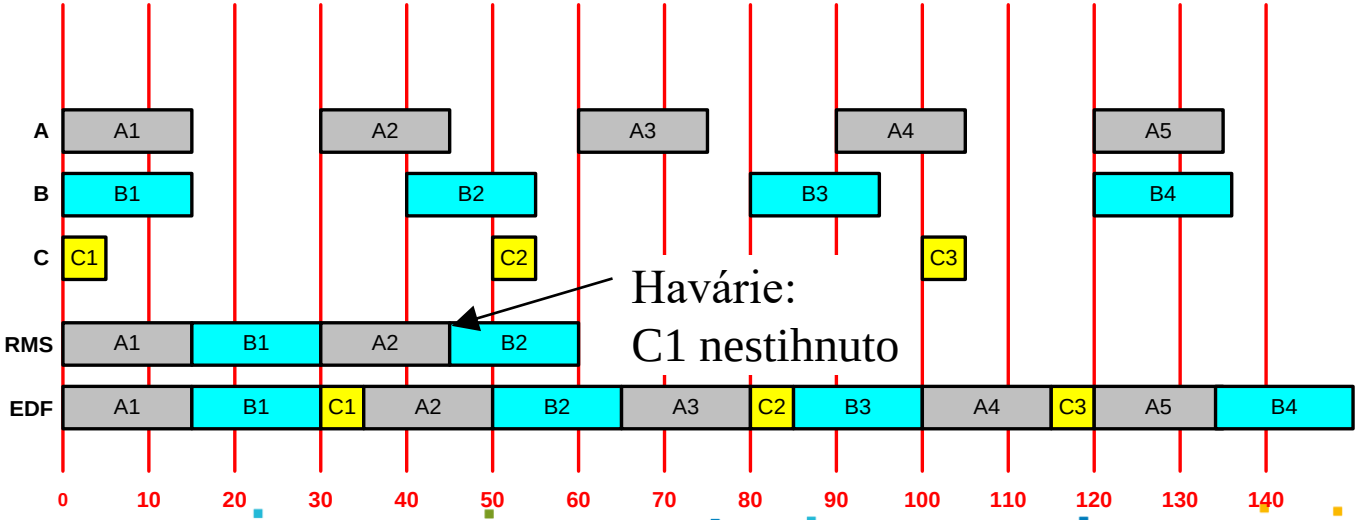
- Oba algoritmy fungují
  - Dokonce chvílemi zbývá volný čas k běhu nějakého procesu „na pozadí“

Příklad 2

- Opět 3 periodické procesy

Proces	Perioda p_i	Procesní čas t_i	t_i / p_i	Suma
A	30	15	0,500	$r = 0{,}975 < 1$
B	40	15	0,375	
C	50	5	0,100	

Plánovatelné



Porovnání RMS a EDF

- RMS:
  - Dobře analyticky zpracovaný algoritmus zaručující dodržení termínů dokončení, pokud při *N* procesech platí (***postačující podmínka***):
  - Používáno v mnoha RT OS
  - Vypracovány i způsoby spolupráce sdílených systémových prostředků
  - Existuje i nutná podmínka (komplikovaná souhra soudělnosti p_i a t_i)

$$r = \sum_{i=1}^{N} \frac{t_i}{p_i} \leq N\left(\sqrt[N]{2} - 1\right);$$

$$\lim_{N \to \infty} N\left(\sqrt[N]{2} - 1\right) = \ln 2 \approx 0.693147$$

N	$N(\sqrt[N]{2}-1)$
2	0,828427
3	0,779763
4	0,756828
5	0,743491
10	0,717734
20	0,705298

- EDF:
  - Při plánování periodických procesů lze dodržet dokončovací termíny i při vytížení téměř 100%
  - Následky přetížení však nejsou známy a nejsou předvídatelné
  - Není známo chování v případech, kdy dokončovací termín a perioda jsou různé

OSTRAVSKÁ UNIVERZITA
PŘÍRODOVĚDECKÁ FAKULTA

Děkuji za pozornost