Univerzita Hradec Králové
Fakulta informatiky a managementu

Přidělování CPU

Mgr. Josef Horálek

FIM UHK

Přidělování CPU

- Přidělování CPU je základ multiprogramového OS
  - pomocí přidělování CPU různým procesům OS zvyšuje výkon výpočetního systému;
- Základní myšlenka multiprogramování je relativně jednoduchá:
  - v jednoduchém OS pokud jediná spuštěná úloha musí čekat je blokováno CPU;
  - v multiprogramovém OS je možno tento čas využít produktivně;
    - v paměti je současně uchováváno více úloh;
    - v okamžiku, kdy musí běžící úloha čekat, je CPU přiděleno jiné;

Základní principy

- CPU cyklus - I/O cyklus
  - běh procesu je složen z cyklu spuštění na CPU a čekání na I/O;
    - probíhající proces přebíhá mezi těmito dvěma cykly;
      - spuštění procesu začíná CPU cyklem
      - následuje I/O cyklus
      - opět CPU cyklus atd.
  - při posledním CPU cyklu dojde k systémovému volání, které vede k ukončení procesu;

FIM UHK

Plánovač CPU

- V okamžiku, kdy by mělo CPU být spuštěným procesem blokováno, OS ho musí přidělit jinému procesu z fronty připravených
  - tento výběr provádí plánovač CPU;
  - fronta připravených může být implementována jako:
    - FIFO;
    - fronta s prioritou;
    - strom;
    - neuspořádaný seznam;
  - každý proces je ve frontě reprezentován zejména svým PCB;

FIM UHK

Preemptivní plánování

- K předělení CPU může dojít kvůli některému z následujících důvodů:
  - jestliže proces přechází ze stavu probíhající do stavu čekající;
    - např. kvůli I/O požadavku nebo kvůli čekání na dokončení synovského procesu;
  - jestliže proces přechází ze stavu probíhající do stavu připraven;
    - např. kvůli přerušení;
  - jestliže proces přechází ze stavu čekající do stavu připraven;
    - např. díky dokončení I/O operace;
  - jestliže je proces ukončen;

FIM UHK

Nepreemptivní plánování

- Je-li v případě nepreemptivního plánování CPU přidělen procesu, ten ho opouští pouze pokud je ukončen nebo čeká
  - např. na I/O operaci;
  - tato metoda plánování je užita v prostředí Microsoft Windows;
  - nepotřebuje totiž speciální hardware jako preemptivní plánování (časovač apod.);

FIM UHK

Dispečer

- Částí OS, která zajišťuje přidělování CPU
  - funkce dispečera jsou:
    - přepínání kontextu;
    - přepínání mezi uživatelskými mody;
    - nalezeni místa, kde byl uživatelský program přerušen a znovu ho spustit;

FIM UHK

Kritéria přidělování CPU

- Různé přidělovací algoritmy mají různé vlastnosti a mohou upřednostňovat třídu procesu před jinými
- Vybíráme-li, který algoritmus užít v určité konkrétní situaci, musíme uvažovat právě vlastnosti různých algoritmů
- Kritéria užitá pro posouzení plánování CPU jsou:
  - využití procesoru;
  - propustnost;
  - doba obrátky;
  - doba čekání;
  - doba odezvy;

FIM UHK

Kritéria přidělování CPU

- Každý OS se snaží
  - maximalizovat využití CPU a propustnost;
  - minimalizovat dobu obrátky, čekání a odezvy;
  - Často výhodnější minimalizovat průměr, než minimum či maximum;
  - v interaktivních systémech je výhodnější minimalizovat rozptyl doby odezvy než její průměr;

First-Come, First-Served Scheduling

- Algoritmus první přišel, první dostal
  - zdaleka nejjednodušší algoritmus plánování CPU;
  - jedná se o „FIFO“ implementaci;
  - proces, který právě požádá o přidělení CPU jej dostane;
  - průměr čekací doby u FCFS algoritmu je však velký;
    - uvažujme sled procesu, který nastal od času 0 s následující délkou CPU cyklu v ms:

Proces	Délka CPU cyklu
P1	24
P2	3
P3	3

First-Come, First-Served Scheduling

= dorazí-li procesy v pořadí P1, P2, P3 a budou obslouženy FCFS algoritmem, výsledkem bude následující diagram:

= dorazí-li procesy v pořadí P2, P3, P1 výsledek bude následující:

First-Come, First-Served Scheduling

- Konvoj efekt
  - procesy čekají na jeden velký proces, který obsadil CPU;
  - jeden proces výrazně orientovaný na CPU;
  - důsledkem konvoj efektu;
    - výrazně nižší využití CPU a I/O zařízení;
  - FCFS je nepreemptivní algoritmus přidělování CPU
    - dostane-li proces CPU, drží si ho do té doby, než je ukončen;
    - nebo než začne provádět nějakou I/O operaci;
      - je nevhodný pro systémy se sdílením času, kde by měl každý uživatel dostat odpověď v přiměřené době;

Shortest-Job-First Scheduling

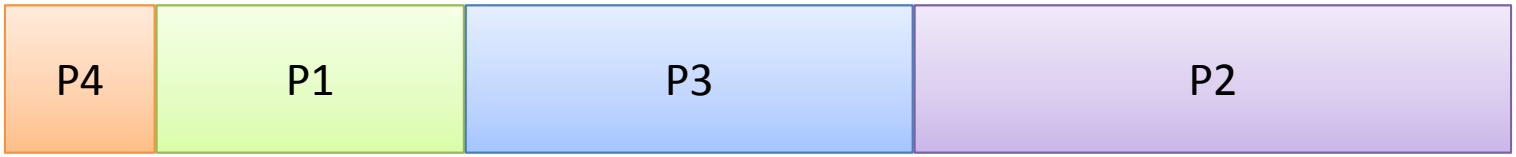
- První jde nejkratší úloha
  - v tomto algoritmu je s každým procesem asociován čas jeho následujícího CPU cyklu;
    - je-li CPU volný, je přidělen procesu s nejkratším následujícím CPU cyklem;
  - jestliže má více procesů stejnou dobu následujícího CPU cyklu, je mezi nimi rozhodnuto algoritmem FCFS;

Shortest-Job-First Scheduling

- Pro příklad uvažujme množinu procesů s uvedenou délkou následujícího CPU cyklu:

Proces	Délka CPU cyklu
P1	6
P2	8
P3	7
P4	3

- Užitím SJF plánování bude procesům přidělen procesor následovně:



FIM UHK

Shortest-Job-First Scheduling

- SJF algoritmus vykazuje minimální průměrnou dobu čekání
  - při přesunutí krátkého procesu před dlouhý se více zkrátí čekací doba krátkého procesu než se prodlouží čekací doba procesu dlouhého
- Reálný problém tohoto algoritmu představuje nutnost vědět dopředu délku následujícího CPU cyklu
- Nemůže být implementován v systémech s krátkodobým plánováním
  - délka dalšího CPU cyklu procesu je odhadnuta jako exponenciální průměr všech předcházejících CPU cyklů procesu;

Shortest-Job-First Scheduling

- SJF algoritmus může být jak preemptivní tak i nepreemptivní
- K výběru dochází, když nový proces dorazí do fronty připravených
  - zatímco předcházející je ještě spuštěný;
    - nový proces může mít kratší CPU cyklus;
  - při preemptivním plánování bude probíhající proces odstaven a CPU přidělen novému procesu;
  - při nepreemptivní plánování algoritmus SJF nechá původní proces dokončit jeho CPU cyklus;

FIM UHK

Preemptivní SJFS

- Jako příklad uvažujme následující 4 procesy s uvedenou délkou jejich následujícího CPU cyklu:

Proces	Délka CPU cyklu
P1	8
P2	4
P3	9
P4	5

- Jestliže procesy dorazí do fronty připravených tak, jak je uvedeno, výsledek preemptivního SJF plánování je vidět na obrázku:

Plánování podle priority

- Plánování podle priority:
  - algoritmus SJF je speciálním případem plánování podle priority
    - principielně toto plánování znamená, že každý proces je asociován se svou prioritou a CPU je přidělen procesu s nejvyšší prioritou;
    - mezi procesy se stejnou prioritou se plánuje FCFS algoritmem;
  - v případě SJF plánování je prioritou **p** převrácená hodnota odhadnuté doby;

FIM UHK

Plánování podle priority

- Priorita může být definována buď interní nebo externí
  - interní priorita užívá měřitelné nebo počitatelné veličiny;
  - externí priorita je nastavována podle vnějších kritérií OS, jako např. důležitost procesu;
  - plánování dle priority může být preemptivní i nepreemptivní;

FIM UHK

Plánování podle priority

- Problémy plánování dle priority jsou:
  - neomezení zablokování (indefinite blocking);
    - procesy s velmi nízkou prioritou by mohli na CPU čekat až neomezeně dlouho;
    - v silně zatíženém výpočetním systému může soustavný tok procesu s vyšší prioritou zabránit procesu s prioritou nižší jakkoli se dostat k CPU;
  - umoření (starvation) procesu;
    - řešením problému neomezeného zablokování procesu je aging;
      - Technika významně zvýšující prioritu procesu, který je v systému dlouhou dobu;
        - např. je-li priorita definována na intervalu 0 - 127, je možné inkrementovat prioritu o 1 každých 15 minut;

FIM UHK

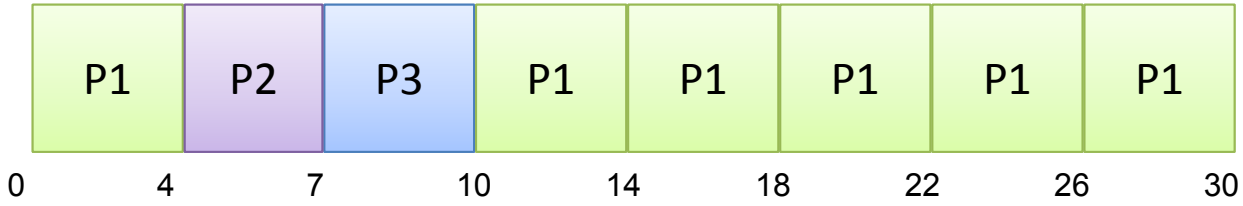
Plánování cyklickou obsluhou

- Plánování cyklickou obsluhou (Round-Robin Scheduling):
  - algoritmus vhodný pro OS se sdílením času
    - podobný algoritmu FCFS, obsahuje však navíc možnost preemptivního plánování;
  - při plánování RR je fronta připravených implementována FIFO frontou procesu a nový proces je vždy zařazen na její konec;
- Nastane jedna ze dvou možností:
  - CPU cyklus procesu může být kratší než časové kvantum;
  - CPU cyklus aktuálně spuštěného procesu je delší než jedno časové kvantum;
    - během vykonávání procesu vyvolá časovač přerušení;

Plánování cyklickou obsluhou

- Průměrná čekací doba při RR plánování je značná.
  - např. délku časového kvanta 4 ms

Proces	Délka CPU cyklu
P1	24
P2	3
P3	3



- Čekací doba procesu P1 je 6 ms, procesu P2 je 4 ms a procesu P3 je 7 ms;
  - průměrná čekací doba je tedy 17/3 = 5.66 ms;
  - RR algoritmus je preemptivní;

Plánování cyklickou obsluhou

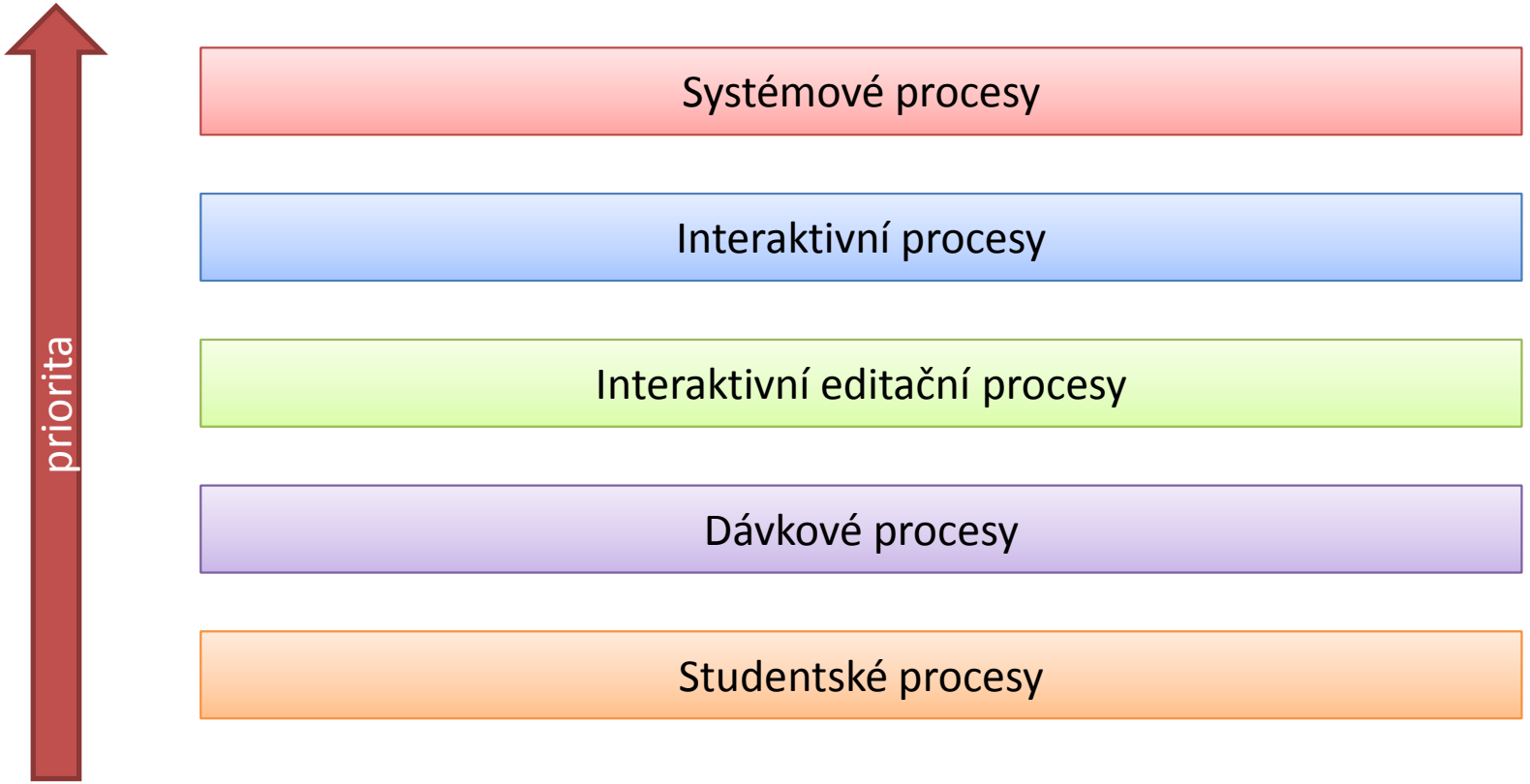
- Efektivita RR plánování závisí na délce časového kvanta
  - je-li časové kvantum příliš velké;
    - RR algoritmus se přeměňuje na FCFS algoritmus se všemi jeho nedostatky
  - je-li časové kvantum příliš malé;
    - stává se RR přístup sdílením procesoru a uživatelsky se zdá, že každý z n procesů je spuštěn na vlastním procesoru s výkonem 1/n oproti skutečnému procesoru systému;
    - při definici časového kvanta je třeba brát v úvahu počet přepínání kontextu, který se zvyšuje se snižující se délkou časového kvanta;

Plánování pomocí více front

- Plánování pomocí více front (Multilevel Queue Scheduling):
  - procesy jsou rozděleny do skupin;
    - jednoduchým příkladem jsou interaktivní procesy a procesy na pozadí;
    - obě skupiny procesů mají různé nároky na doby odezvy;
      - měly by tedy být odlišně plánovány
    - procesy na popředí mají vyšší prioritu (externí) než procesy na pozadí;
  - každá fronta (skupina) má svůj vlastní plánovací algoritmus;
    - musí existovat plánování mezi jednotlivými frontami;
      - většinou implementované jako preemptivní plánování s pevnou prioritou;
      - další možností je definovat časové intervaly přidělení CPU jednotlivým frontám;
      - každá fronta dostane určitou část času CPU, během kterého může plánovat procesy v ní uložené;

FIM UHK

Plánování pomocí více front

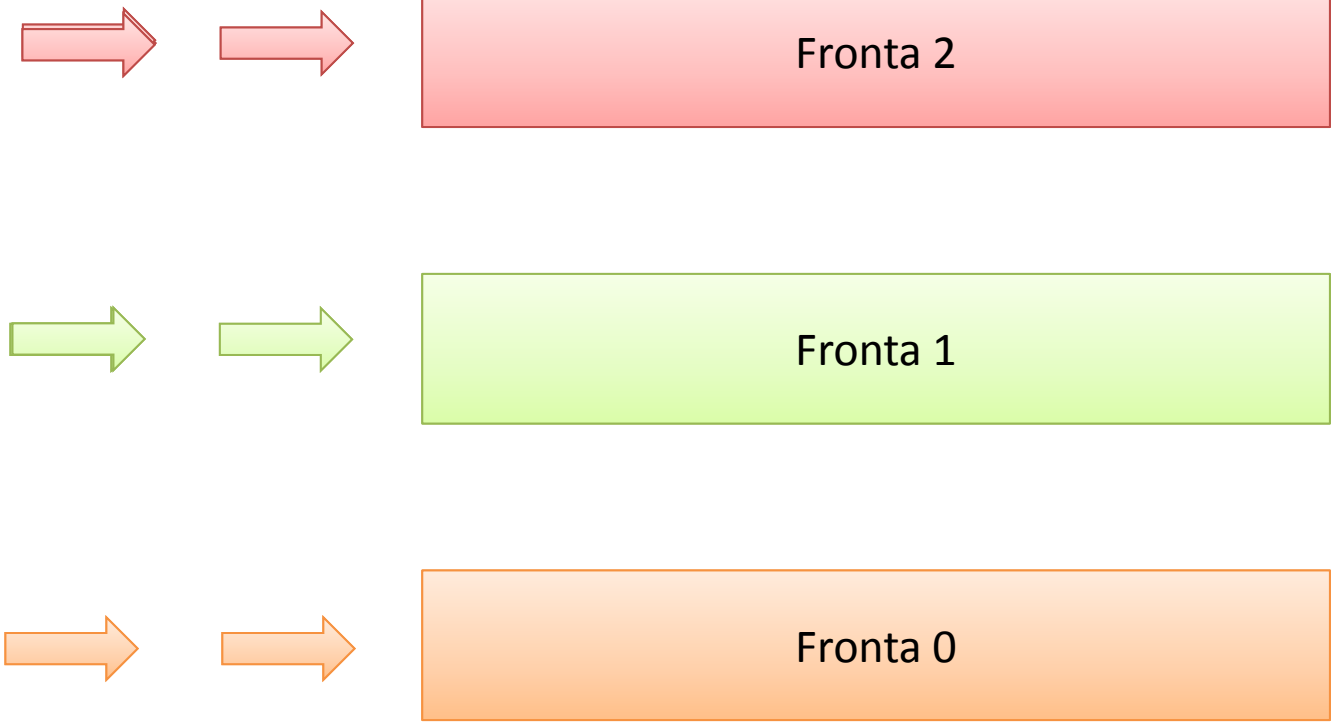


Plánování pomocí více front se zpětnou vazbou

- Plánování pomocí více front se zpětnou vazbou (Multilevel Feedback Queue Scheduling):
  - v systému plánování MQS je každému procesu přidělena fronta, ve které bude čekat na CPU již v okamžiku vytvoření procesu;
  - procesy se mezi frontami nemohou přesouvat;
  - tento systém snižuje režii plánování, ale není příliš flexibilní;
- MFQS umožňuje procesům přesouvat se mezi frontami;
  - důvod je oddělit procesy s různou charakteristikou CPU cyklu;
- schéma ponechává vysokou prioritu interaktivním a I/O procesům;

FIM UHK

Plánování pomocí více front se zpětnou vazbou



Plánování pomocí více front se zpětnou vazbou

- Plánování pomocí více front se zpětnou vazbou je jedním z nejvýznamnějších algoritmů plánování CPU
- Může být konfigurován pro libovolný specifický systém, protože je velmi komplexní

Univerzita Hradec Králové
Fakulta informatiky a managementu

Děkuji za pozornost…