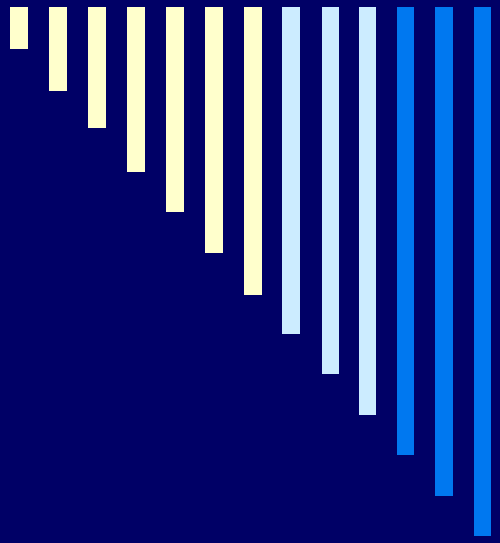
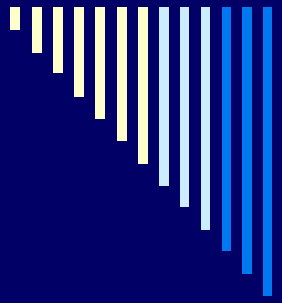


07.
Plánování procesů
Deadlock

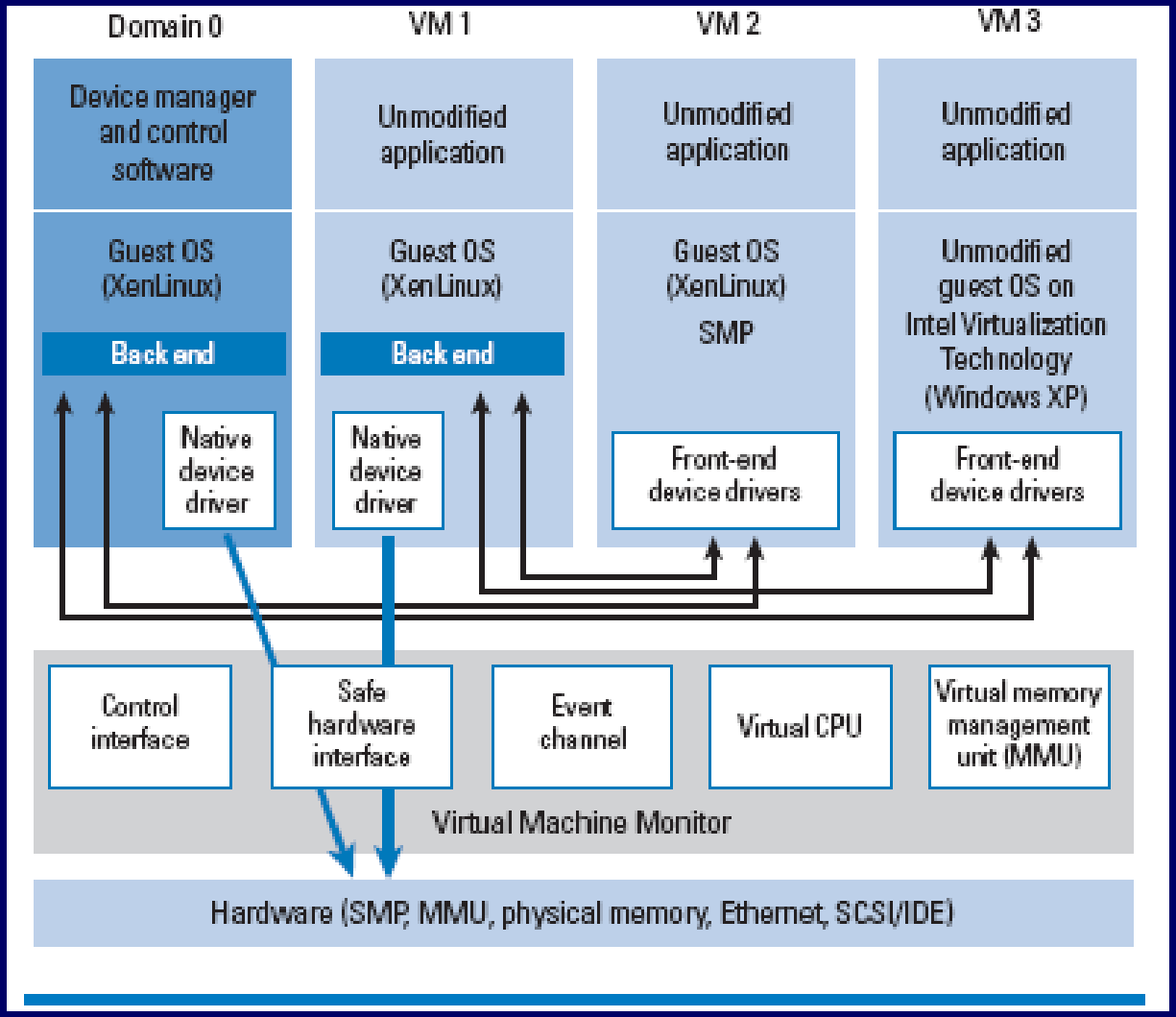
ZOS 2006, L. Pešička

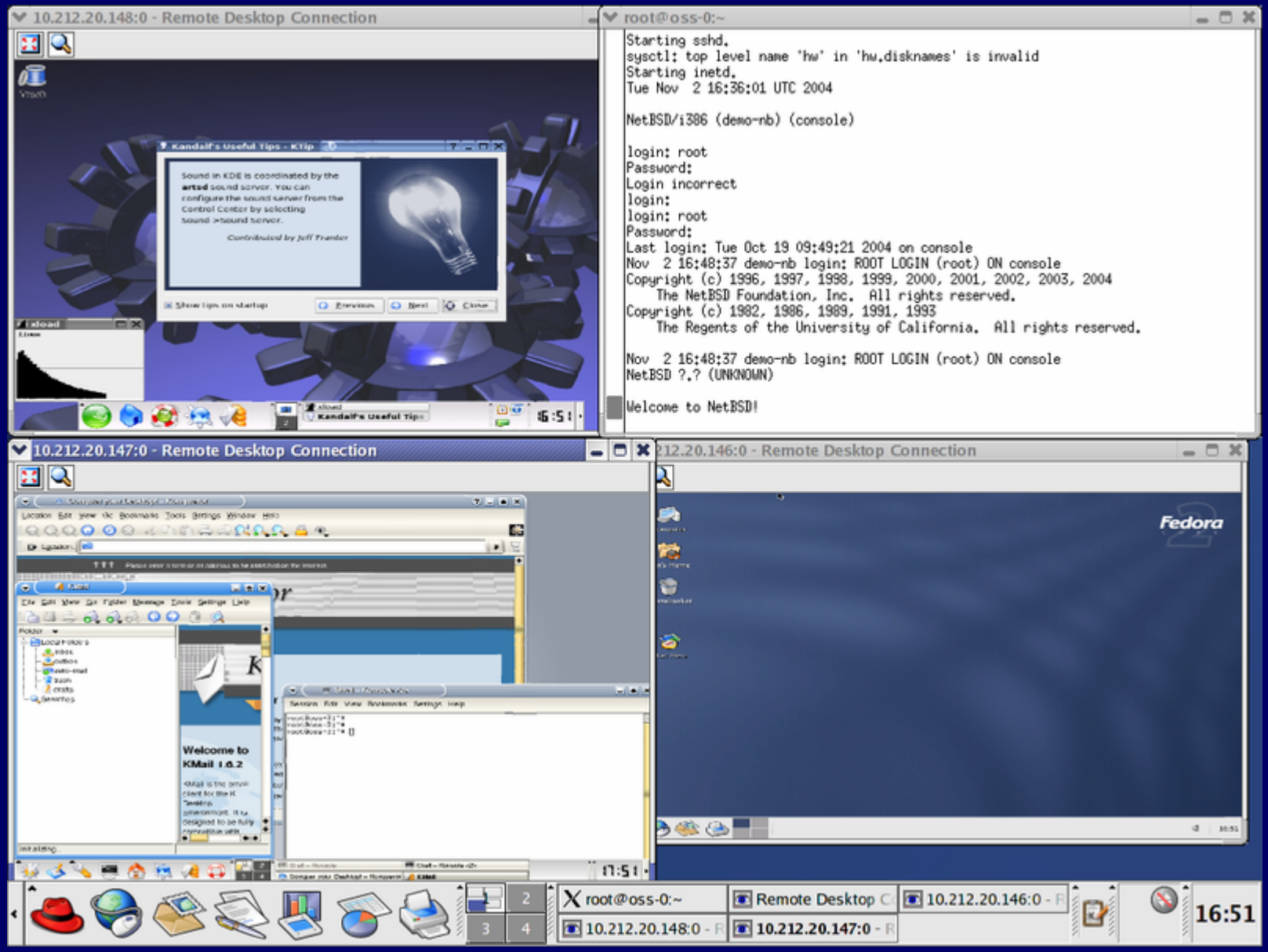


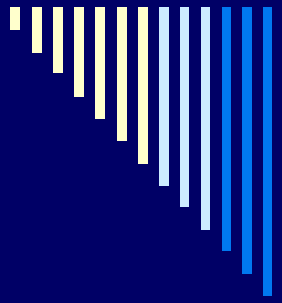
Pozvánka na přednášku
Xen na ZČU

- architektura
- migrace virtuálního stroje
- reálná konfigurace na ZČU
- praktická ukázka konfigurace a instalace virtuálního stroje na xenu včetně migrace

- 22.11. (St) , 15:30, UI113 (zasedačka CIV)

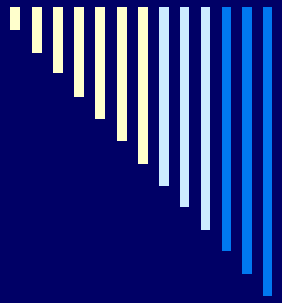

Domain 0
VM 1
VM 2
VM 3
Device manager and control software
Unmodified application
Unmodified application
Unmodified application
Guest OS (XenLinux)
Guest OS (XenLinux)
Guest OS (XenLinux) SMP
Unmodified guest OS on Intel Virtualization Technology (Windows XP)
Back end
Back end
Native device driver
Native device driver
Front-end device drivers
Front-end device drivers
Control interface
Safe hardware interface
Event channel
Virtual CPU
Virtual memory management unit (MMU)
Virtual Machine Monitor
Hardware (SMP, MMU, physical memory, Ethernet, SCSI/IDE)


10.212.20.148:0 - Remote Desktop Connection
root@oss-0:~
Starting sshd.
sysctl: top level name 'hw' in 'hw.disknames' is invalid
Starting inetd.
Tue Nov 2 16:36:01 UTC 2004
NetBSD/i386 (demo-nb) (console)
login: root
Password:
Login incorrect
login:
login: root
Password:
Last login: Tue Oct 19 09:49:21 2004 on console
Nov 2 16:48:37 demo-nb login: ROOT LOGIN (root) ON console
Copyright (c) 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003, 2004
The NetBSD Foundation, Inc. All rights reserved.
Copyright (c) 1982, 1986, 1989, 1991, 1993
The Regents of the University of California. All rights reserved.
Nov 2 16:48:37 demo-nb login: ROOT LOGIN (root) ON console
NetBSD ?.? (UNKNOWN)
Welcome to NetBSD!
10.212.20.147:0 - Remote Desktop Connection
212.20.146:0 - Remote Desktop Connection
Fedora
Welcome to KMail 1.6.2
root@oss-0:~
Remote Desktop C
10.212.20.146:0 - R
10.212.20.148:0 - R
10.212.20.147:0 - R
16:51



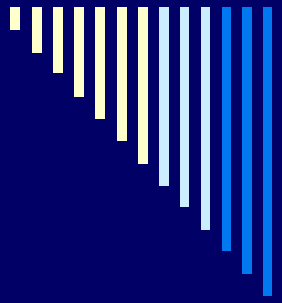
Plánování procesů

- Plánování procesů v dávk. systémech - minule
- Plánování procesů v interaktivních systémech
- Příklad – Windows 2000
- Plánování ve víceprocesorových systémech
- Plánování v systémech reálného času
- Plánování procesů x plánování vláken



Plánování procesů v interaktivních systémech

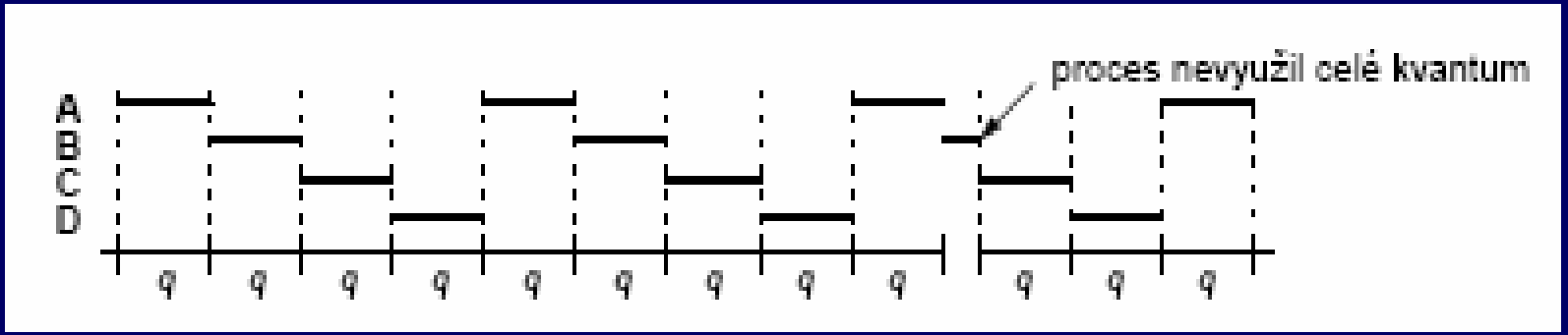
- potřeba docílit, aby proces neběžel „příliš dlouho“
  - možnost obsloužit další procesy
- každý proces – jedinečný a nepredikovatelný
  - nelze říct, jak dlouho poběží, než se zablokuje (nad I/O, semaforem, ...)
- vestavěný systémový časovač v počítači
  - provádí pravidelně přerušení (tiky časovače, clock ticks)
  - vyvolá se obslužný podprogram v jádře
  - rozhodnutí, zda proces bude pokračovat, nebo se spustí jiný (preemptivní plánování)

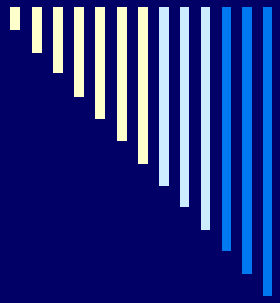


Algoritmus cyklické obsluhy – Round Robin (RR)

- jeden z nejstarších a nejpoužívanějších
- každému procesu přiřazen časový interval
  - časové kvantum, po které může běžet
- proces běží na konci kvanta
  - preemce, naplánován a spuštěn další připravený proces
- proces skončí nebo se zablokuje před uplynutím kvanta
  - stejná akce jako v předchozím bodě ☺

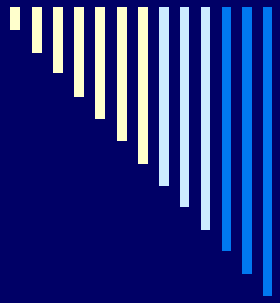
Round Robin





Round Robin

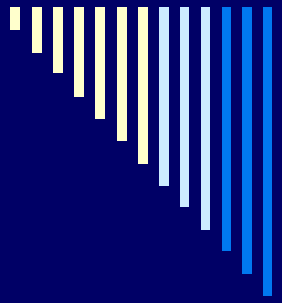
- jednoduchá implementace plánovače
  - plánovač udržuje seznam připravených procesů
  - vypršení kvanta / zablokování – vybere další proces



Obslužný program přerušení

- v jádře

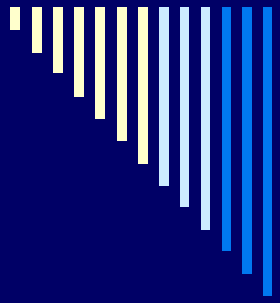
- nastavuje interní časovače systému
- shromažďuje statistiky systému
  - kolik času využíval CPU který proces, ...
- po uplynutí kvanta (resp. v případě potřeby) zavolá plánovač



1 kvantum – více přerušení časovače

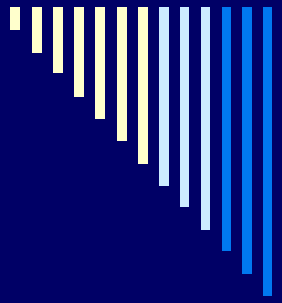
- Časovač může proces v průběhu časového kvanta přerušit vícekrát.

- přerušení 100x za sekundu (příklad)
  - 10 ms mezi přerušeními
- pokud kvantum 50 ms
  - přeplánování každý pátý tik



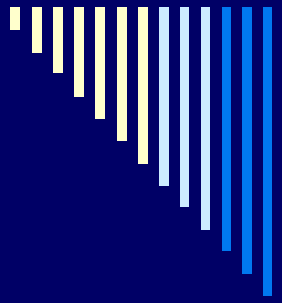
vhodná délka časového kvanta

- krátké
  - přepnutí procesů chvíli trvá (uložení a načtení registrů, přemapování paměti, ...)
  - přepnutí kontextu 1ms, kvantum 4ms – 20% velká režie
- dlouhé
  - vyšší efektivita; kvantum 1s – 1% režie
  - pokud kvantum delší než průměrná doba držení CPU procesem – preempce je třeba zřídka
  - problém interaktivních procesů – 10 uživatelů stiskne klávesu, odezva posledního procesu až 10s



vhodná délka kvanta - shrnutí

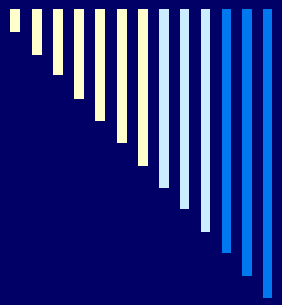
- krátké kvantum – snižuje efektivitu (režie)
- dlouhé – zhoršuje dobu odpovědi na interaktivní požadavky
- kompromis ☺
- pro algoritmus cyklické obsluhy obvykle 20 až 50 ms
- kvantum nemusí být konstantní
  - změna podle zatížení systému
- pro algoritmy, které se lépe vypořádají s interaktivními požadavky lze kvantum delší – 100 ms



Problém s algoritmem cyklické obsluhy

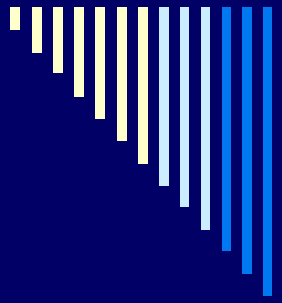
- v systému výpočetně vázané i I/O vázané úlohy
- výpočetně vázané – většinou kvantum spotřebují
- I/O vázané – pouze malá část kvanta a zablokují se
- výpočetně vázané – získají nespravedlivě vysokou část času CPU

- modifikace VRR (Virtual RR, 1991)
  - procesy po dokončení I/O mají přednost před ostatními



Prioritní plánování

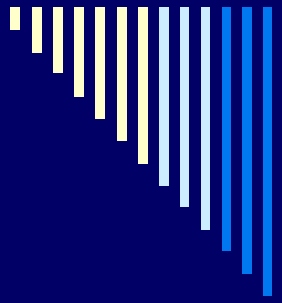
- předpoklad RR: všechny procesy stejně důležité
- ale:
  - vyšší priorita zákazníkům, kteří si „připlatí“
  - interaktivní procesy vs. procesy běžící na pozadí (odesílání pošty)
- prioritu lze přiřadit staticky nebo dynamicky:
- staticky
  - při startu procesu, např. Linux – nice
- dynamicky
  - přiradit I/O větší prioritu, použití CPU a zablokování



Priorita

- obě složky – výsledná jejich součtem
- statická (při startu procesu) a dynamická (chování procesu v poslední době)

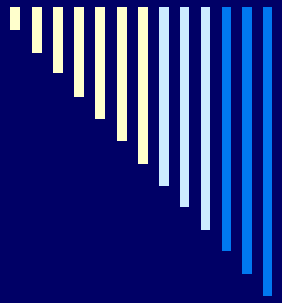
- kdyby pouze statická a plánování jen podle priorit – běží pouze připravené s nejvyšší prioritou
- plánovač snižuje dynamickou prioritu běžícího procesu při každém tiku časovače; klesne pod prioritu jiného - přeplánování



Dynamická priorita

- V kvantově orientovaných plánovacích algoritmech:

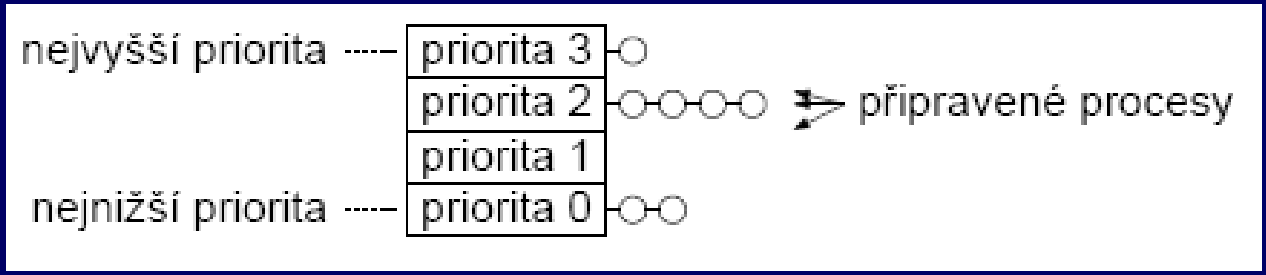
- dynamická priorita 1 / f
- f – velikost části kvanta, kterou proces naposledy použil
- zvýhodní I/O vázané x CPU vázaným



Spojení cyklického a prioritního plánování

- prioritní třídy
  - v každé procesy se stejnou prioritou
- prioritní plánování mezi třídami
- cyklická obsluha uvnitř třídy
- obsluhovány jsou pouze připravené procesy v nejvyšší neprázdné prioritní třídě

Prioritní třídy



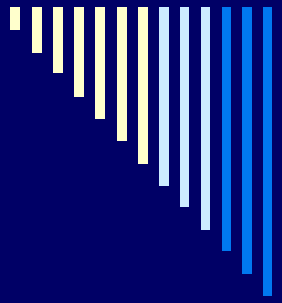
4 prioritní třídy

dokud procesy v třídě 3 – spustit cyklicky každý na 1 kvantum

pokud třída 3 prázdná – totéž pro třídu 2

jednou za čas – přepočítání priorit

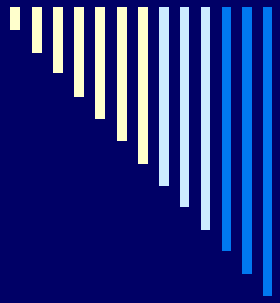
procesům, které využívaly CPU se sníží priorita



Prioritní třídy

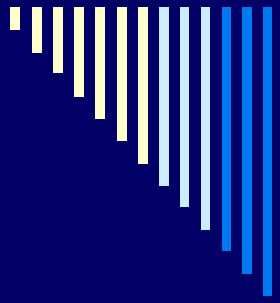
- dynamické přiřazování priority
  - dle využití CPU v poslední době
  - priorita procesu
    - snižuje se při běhu
    - zvyšuje při nečinnosti
- cyklické střídání procesů

- OS typu Unix
  - 30 až 50 prioritních tříd



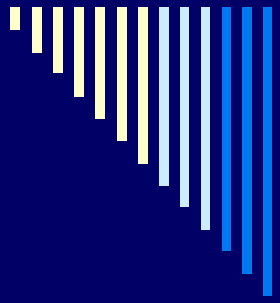
Plánovač spravedlivého sdílení

- problém:
  - čas přidělován každému procesu nezávisle
  - uživatel – více procesů -> dostane více času celkově
- spravedlivé sdílení
  - přidělovat čas každému uživateli (či jinak definované skupině procesů) proporcionálně, bez ohledu na to, kolik má procesů
  - N uživatelů, každý dostane 1/N času



Spravedlivé sdílení

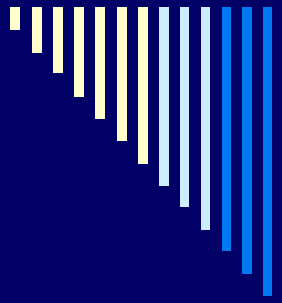
- nová položka priorita skupiny spravedlivého plánování
  - pro každého uživatele
- obsah položky
  - započítává se do priority každého procesu uživatele
  - odráží poslední využití procesoru všemi procesy uživatele



Spravedlivé sdílení - implementace

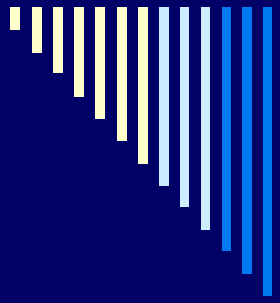
- každý uživatel – položka g
- obsluha přerušení – inkrementuje g uživatele, kterému patří běžící proces
- jednou za sekundu rozklad: g=g/2

- priorita P (p,g) = p – g
- pokud procesy uživatele využívaly CPU v poslední době – položka g je vysoká



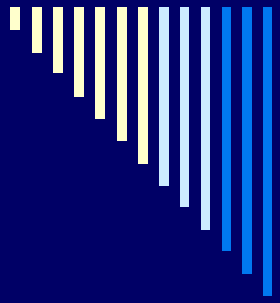
Plánování pomocí loterie

- Lottery Scheduling (Waldspurger & Weihl, 1994)
- cílem – poskytnout proces. příslušnou proporci času CPU
- základní princip:
  - procesy obdrží tikety (losy)
  - plánovač vybere náhodně jeden tiket
  - vítězný proces obdrží cenu – 1 kvantum času CPU
  - důležitější procesy – více tiketů, aby se zvýšila šance na výhru (celkem 100 losů, proces má 20 – v dlouhodobém průměru dostane 20% času)



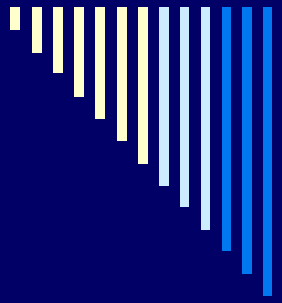
Loterie - výhody

- řešení problémů, v jiných plán. algoritmech obtížné
- spolupracující procesy – mohou si předávat losy
  - klient posílá zprávu serveru a blokuje se
  - může serveru propůjčit všechny své tikety
  - po vykonání požadavku server tikety vrátí
  - nejsou-li požadavky, server žádné tikety nepotřebuje



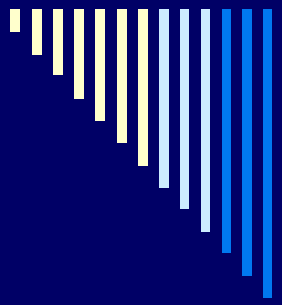
Loterie - výhody

- rozdělení času mezi procesy v určitém poměru
  - neplatí u prioritního plánování, co je to že má proces prioritu např. 30?
  - proces – tickety – šance vyhrát
- relativně nový algoritmus, zatím málo reálných systémů



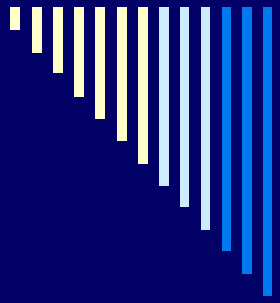
Shrnutí

Algoritmus	Rozhodovací mód	Prioritní funkce	Rozhodovací pravidlo
RR	Preemptivní vyprš. kv.	P() = 1	cyklicky
prioritní	Preemptivní P jiný > P	Viz text	Náhodně, cyklicky
spravedlivé	Preemptivní P jiný > P	P(p,g)=p-g	cyklicky
loterie	Preemptivní vyprš. kv.	P() = 1	Dle výsledku loterie



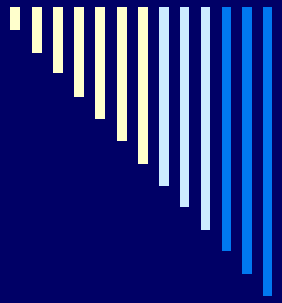
Příklad – Windows 2000

- 32 prioritních úrovní, 0 až 31 (nejvyšší)
- pole 32 položek
  - každá položka – ukazatel na seznam připravených procesů
- plánovací algoritmus – prohledává pole od 31 po 0
  - nalezne neprázdnou frontu
  - naplánuje první proces, nechá ho běžet 1 kvantum
  - po uplynutí kvanta – proces na konec fronty na příslušné prioritní úrovni



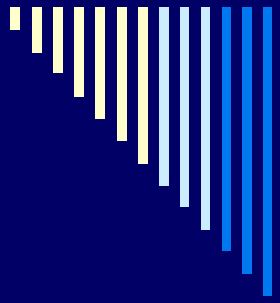
Win 2000 – skupiny priorit

priorita	popis
0	Nulování stránek pro správce paměti
1 .. 15	Obyčejné procesy
16 .. 31	Systémové procesy



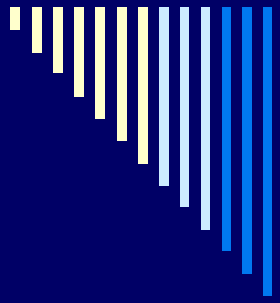
Win 2000 - priority

- 0 .. pokud není nic jiného na práci
- 1 .. 15 – obyčejné procesy
- aktuální priorita – <bázová, 15>
- bázová priorita – základní, může ji určit uživatel voláním *SetPriorityClass*
- aktuální priorita se mění – viz dále
- procesy se plánují přísně podle priorit, tj. obyčejné pouze pokud není žádný systémový proces připraven



Win 2000 – změna akt. priority

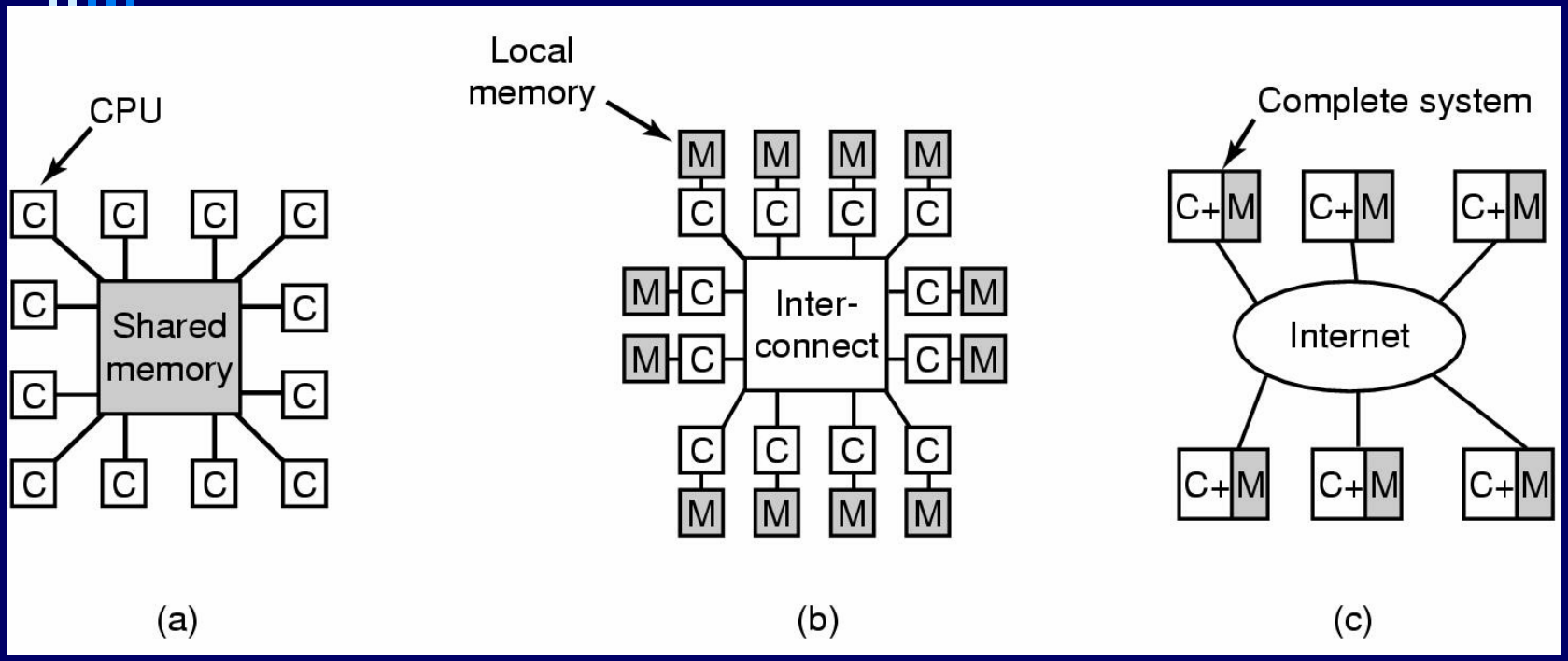
- dokončení I/O zvyšuje prioritu o
  - 1 – disk, 2 – sériový port, 6 – klávesnice, 8 – zvuková karta
- vzbuzení po čekání na semafor, mutex zvýší o
  - 2 - pokud je proces na popředí
    (řídí okno, do kterého je posílán vstup z klávesnice)
  - 1 – jinak
- proces využil celé kvantum
  - sníží se priorita o 1
- proces neběžel dlouhou dobu
  - na 2 kvanta priorita zvýšena na 15 (zabránit inverzi priorit)



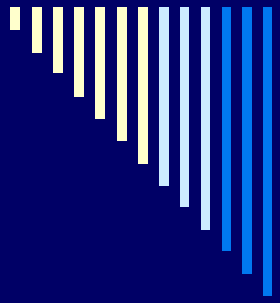
Plánování – víceprocesorové stroje

- nejčastější architektura
  - těsně vázaný symetrický multiprocesor
  - procesory jsou si rovné, společná hlavní paměť
- Přiřazení procesů procesorům – ukázka
  - Permanentní přiřazení
    - Menší režie, některá CPU mohou zahálet
    - Afinita procesu k procesoru, kde běžel naposledy
    - Někdy procesoru přiřazen jediný proces – RT procesy
  - Společná fronta připravených procesů
    - Plánovány na libovolný procesor

Multiprocessor Systems

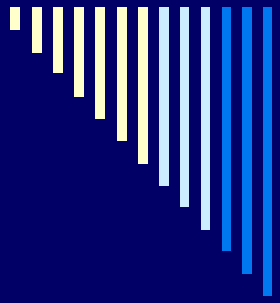


- Continuous need for faster computers
  - shared memory model
  - message passing multiprocessor
  - wide area distributed system



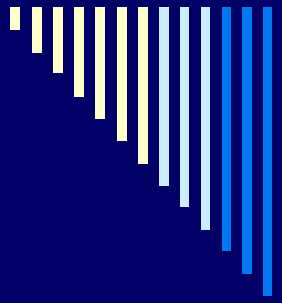
Víceprocesorové stroje

- Plánování vláken
  - Některé paralelní aplikace – podstatně větší výkonnost, pokud jejich vlákna běží současně
    - Zkrátí se vzájemné čekání vláken
  - V současné době – intenzivní výzkum v oblasti OS



Plánování v systémech reálného času

- Stručně – charakteristika RT systémů
  - RT procesy řídí nebo reagují na události ve vnějším světě
  - Správnost závisí nejen na výsledku, ale i na čase, ve kterém je výsledek vyprodukován
  - S každou podúlohou – sdružit deadline – čas kdy musí být spuštěna nebo dokončena
  - Hard RT – času musí být dosaženo
  - Soft RT – dosažení deadline je žádoucí



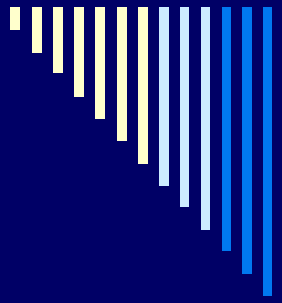
Systémy RT

- Podúlohy procesu (události, na které se reaguje)
  - Aperiodické – nastávají nepredikovatelně
  - Periodické – v pravidelných intervalech
- Zpracování události vyžaduje čas
  - Pokud je možné všechny včas zpracovat
    - systém je plánovatelný (schedulable)

Plánovatelné RT systémy

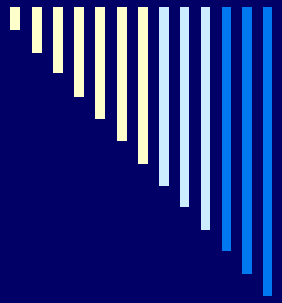
- Je dáno
  - *m* – počet periodických událostí
  - výskyt události *i* s periodou P_i vyžadující C_i sekund
- Zátěž lze zvládnout, pokud platí:

$$\sum_{i=1}^{m} \frac{C_i}{P_i} \leq 1$$



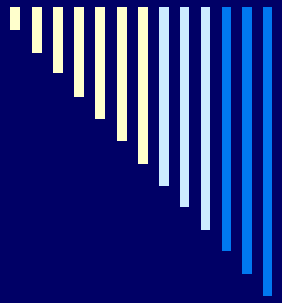
Plánovací algoritmy v RT

- Statické nebo dynamické
- Statické
  - Plánovací rozhodnutí před spuštěním systému
  - Předpokládá dostatek informací o vlastnostech procesů
- Dynamické
  - Za běhu
  - Některé algoritmy provedou analýzu plánovatelnosti, nový proces přijat pouze pokud je výsledek plánovatelný



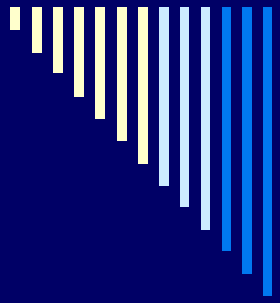
Vlastnosti současných RT

- Malá velikost OS – omezená funkčnost
- Snaha spustit RT proces co nejrychleji
  - Rychlé přepínání mezi procesy nebo vlákny
  - Rychlá obsluha přerušení
  - Minimalizace intervalů, kdy je přerušení zakázáno
- Multitasking + meziprocesová komunikace (semafory, signály, události)
- Primitiva pro zdržení procesu o zadaný čas, čítače časových intervalů
- Někdy rychlé sekvenční soubory (viz později)



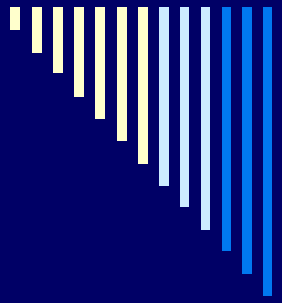
Plánování procesů a vláken

- Plánování procesů – vždy součást OS
- Plánování vláken
  - Běh vláken plánuje OS
    - Kernel-level threads
  - Běh vláken plánován uživatelským procesem
    - User-level threads
    - OS o existenci vláken nic neví



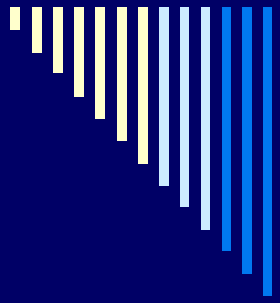
Plánování vláken

- Vlákna plánována OS
  - Stejné mechanismy a algoritmy jako pro plánování procesů
  - Často plánována bez ohledu, kterému procesu patří (proces 10 vláken, každé obdrží časové kvantum)



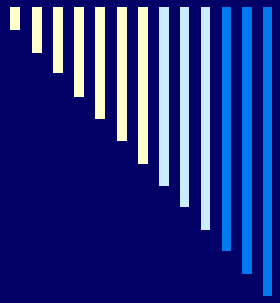
Plánování vláken

- Vlákna plánována uvnitř procesu
  - Běží v rámci času, který je přidělen procesu
  - Přepínání mezi vlákny – systémová knihovna
  - Pokud OS neposkytuje procesu pravidelné “přerušení“, tak pouze nepreemtivní plánování
  - Obvykle algoritmus RR nebo prioritní plánování
  - Menší režie oproti kernel-level threads, menší možnosti
- Windows 2000 a Linux – vlákna plánována jádrem
- Některé varianty UNIXu – user-level threads



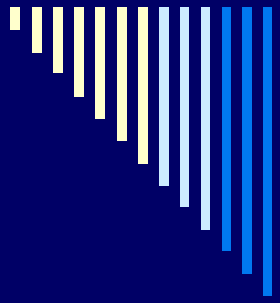
Dispatcher

- Dispatcher
  - Modul, který předá řízení CPU procesu vybraným short term plánovačem
- Provede:
  - Přepnutí kontextu
  - Přepnutí do uživatelského modu
  - Skok na danou instrukci v uživatelském procesu
- Co nejrychlejší, vyvolán během každého přepnutí procesů



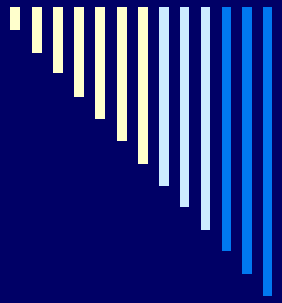
Scheduler – protichůdné požadavky

- příliš časté přepínání procesu – režie
- málo časté – pomalá reakce systému
- čekání na diskové I/O, data ze sítě – probuzen a brzy (okamžitě) naplánován – pokles přenosové rychlosti
- víceprocesor – pokud lze, nestřídat procesory
- nastavení priority uživatelem



Proces – stav blokovaný (Unix)

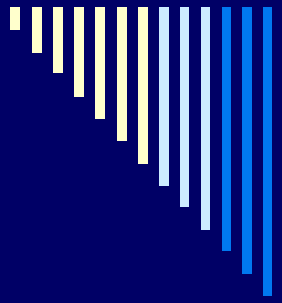
- čeká na událost – ve frontě
- přerušitelné signálem (terminál, sockety, pipes)
  - procesy označené s
    - signál – syscall se zruší – návrat do userspace
    - obsluha signálu
    - znovu zavolá přerušené syst. volání (pokud požadováno)
- nepřerušitelné
  - procesy označené D
  - operace s diskem – skončí v krátkém čase
- plánovač mezi nimi nerozlišuje



Poznámka – vyhladovění procesu

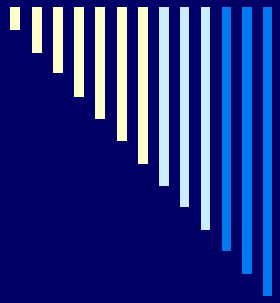
- Ukázka ze života

- V roce 1973 na MITU shut down stroje IBM 7094
- Nalezen proces, který nebyl spuštěn od roku 1967



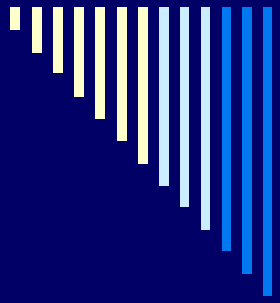
Poznámka - simulace

- Trace tape – monitorujeme běh reálného systému, zaznamenáváme posloupnost událostí
- Tento záznam použijeme pro řízení simulace
- Lze využít pro porovnávání algoritmů
- Trace tape – nutno uložit velké množství dat



Uvíznutí (deadlock)

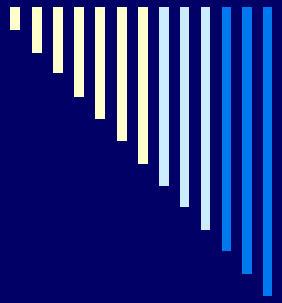
- Příklad:
- Naivní večeřící filozofové – vezmou pravou vidličku, ale nemohou vzít levou (už je obsazena)
- Uvíznutí (deadlock); zablokování



Uvíznutí – alokace I/O zařízení

- Výhradní alokace I/O zařízení
- Vypalovačka CD (R), scanner (S)
- Procesy A, B – naskenovat a zapsat na vypalovačku

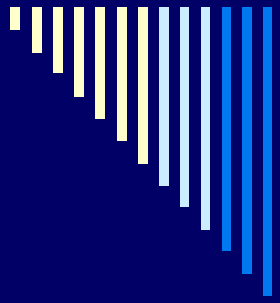
- A žádá R a dostane, B žádá S a dostane
- A žádá S a čeká, B žádá R a čeká -- uvíznutí



Uvíznutí – zamykání záznamů v databázi, semafory

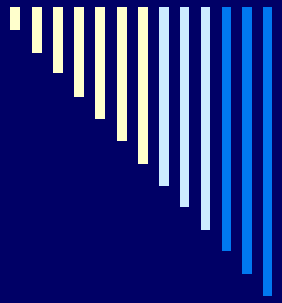
- Dva procesy A, B
  požadují přístup k záznamům v databázi
- A zamkne R, B zamkne S, …

- Semafory R a S, R=1, S=1
- A provede P(R), B provede P(S),
- Co dále, aby deadlock?



Zdroje

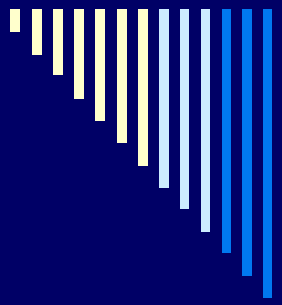
- přeplánovatelné (preemtable)
  - lze je odebrat procesu bez škodlivých efektů
- nepřeplánovatelné (nonpremeptable)
  - proces zhavaruje, pokud jsou mu odebrány



Zdroje

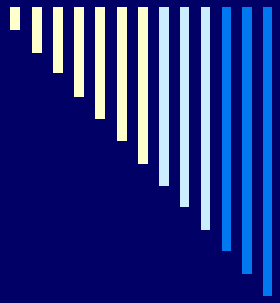
- Sériově využitelné zdroje
  - Proces zdroj alokuje, používá, uvolní
- Konzumovatelné zdroje
  - Např. zprávy, které produkuje jiný proces
  - Viz producent – konzument

  - Také uvíznutí
  - Proces A: … receive (B,R); send (B, S); ..
  - Proces B: … receive (A,S); send (A, R); ..
- Dále bude o seriově využitelných zdrojích



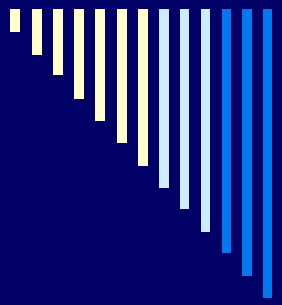
Více zdrojů stejného typu

- Některé zdroje – více exemplářů
- Proces žádá zdroj daného typu – jedno který dostane
- Např. bloky disku pro soubor, paměť, …
- Př. 5 zdrojů a dva procesy A,B
  - A požádá o dva zdroje, dostane (zbydou 3)
  - B požádá o dva zdroje, dostane (zbude 1)
  - A žádá o další dva, nejsou (je jen 1), čeká
  - B žádá o další dva, nejsou, čeká – nastalo uvíznutí
- Zaměříme se na situace, kdy 1 zdroj každého typu



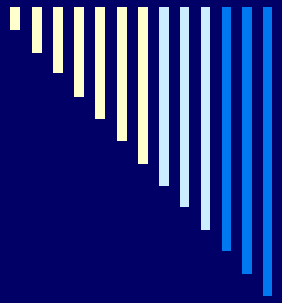
Práce se zdrojem

- Žádost (request)
  - Uspokojena bezprostředně nebo proces čeká
  - Systémové volání
- Použití (use)
  - Např. tisk na tiskárně
- Uvolnění (release)
  - Proces uvolní zdroj
  - Systémové volání



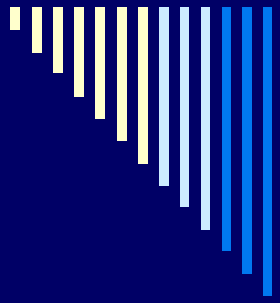
Uvíznutí - definice

- Obecný termín zdroj – zařízení, záznam, …
- V množině procesů nastalo uvíznutí, jestliže každý proces množiny čeká na událost, kterou může způsobit jiný proces množiny
- Všichni čekají – nikdo událost nevygeneruje, nevzbudí jiný proces



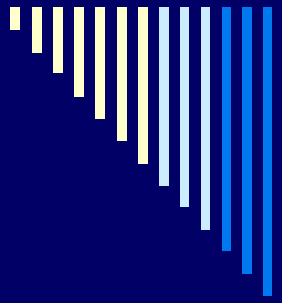
Podmínky vzniku uvíznutí

- Coffman, 1971
- 1. vzájemné vyloučení
  - Každý zdroj je buď dostupný nebo je výhradně přiřazen právě jednomu procesu
- 2. hold and wait
  - Proces držící výhradně přiřazené zdroje může požadovat další zdroje



Podmínky vzniku uvíznutí

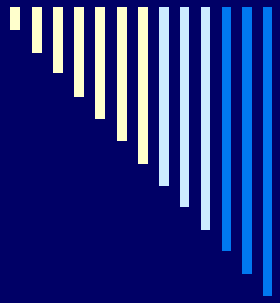
- 3. nemožnost odejmutí
  - Jednou přiřazené zdroje nemohou být procesu násilně odejmuty (proces je musí sám uvolnit)
- 4. cyklické čekání
  - Musí být cyklický řetězec 2 nebo více procesů, kde každý z nich čeká na zdroj držený dalším členem



Vznik uvíznutí - poznámky

- Pro vznik uvíznutí – musejí být splněny všechny 4 podmínky
  - 1. až 3. předpoklady, za nich je definována 4. podmínka
- Pokud jedna z podmínek není splněna, uvíznutí nenastane

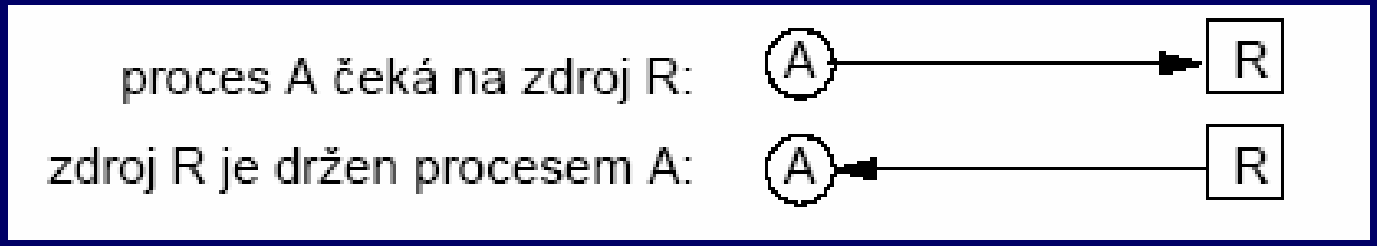
- Viz příklad s CD vypalovačkou
  - Na CD může v jednu chvíli zapisovat pouze 1 proces
  - CD rekorder není možné zapisovacímu procesu odejmout



Modelování uvíznutí

- Graf alokace zdrojů
- 2 typy uzlů
  - Proces – zobrazujeme jako kruh
  - Zdroj – jako čtverec
- Význam hran
  - Hrana od zdroje k procesu:
    - zdroj držen procesem
  - Hrana od procesu ke zdroji:
    - proces blokován čekáním na zdroj

Modelování uvíznutí

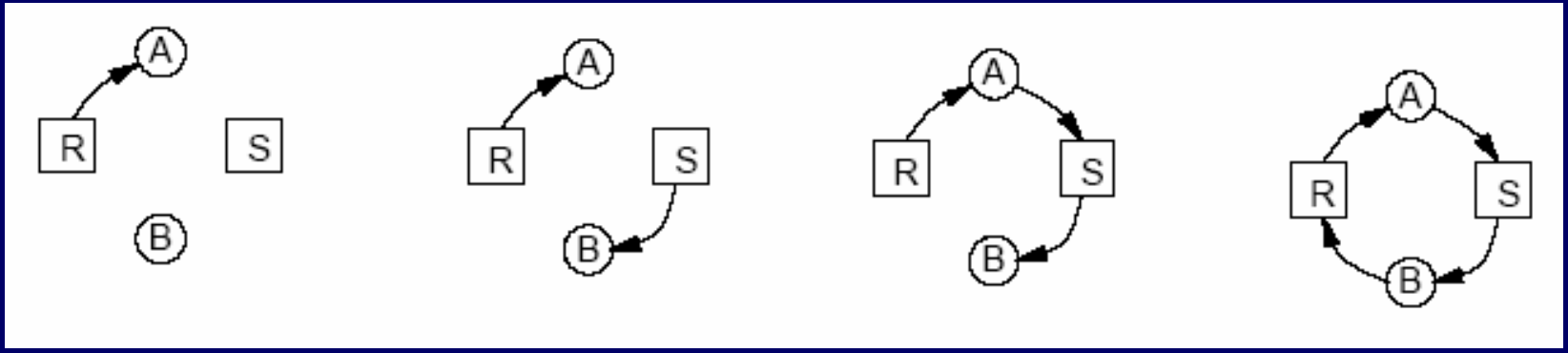


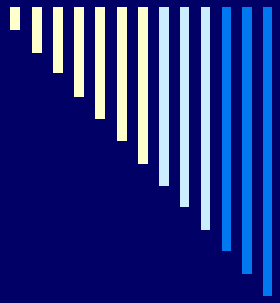
Cyklus v grafu – nastalo uvíznutí

Uvíznutí se týká procesů a zdrojů v cyklu

Uvíznutí

- Rekorder R a scanner S; dva procesy A,B
- A žádá R dostane, B žádá S dostane
- A žádá S a čeká, B žádá R a čeká - uvíznutí





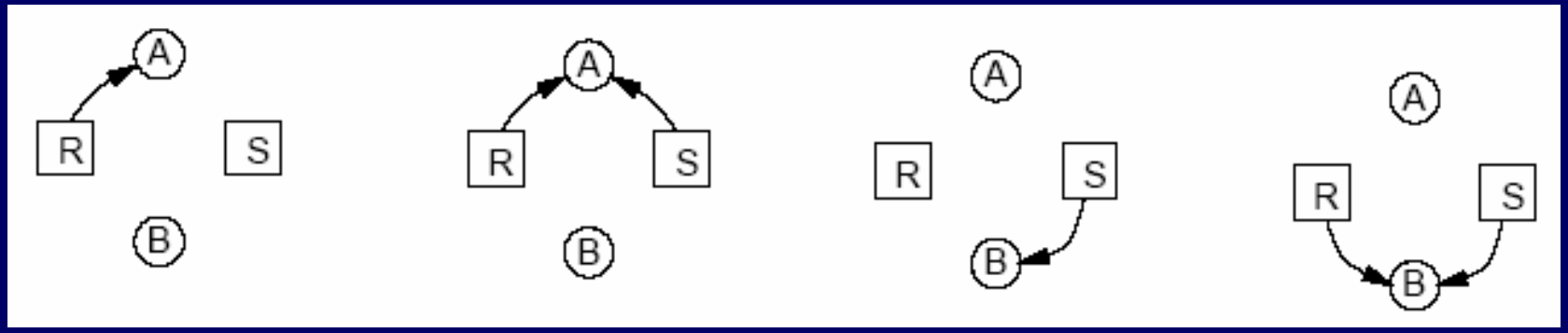
Uvíznutí - poznámky

- Cyklus v grafu – nutnou a postačující podmínkou pro vznik uvíznutí

- Závisí na pořadí vykonávání instrukcí procesů
- Nejprve alokace a uvolnění zdrojů procesu A, potom B
  - Uvíznutí nenastane

Uvíznutí - poznámky

- A žádá R a S, oba dostane, A oba zdroje uvolní
- B žádá S a R, oba dostane, B oba zdroje uvolní
- Nenastane uvíznutí
- Při některých běžích nemusí uvíznutí nastat – hůře se hledá chyba



Uvíznutí – pořadí alokace

- Pokud bychom napsali procesy A,B tak, aby oba žádaly o zdroje R a S ve stejném pořadí – uvíznutí nenastane
- A žádá R a dostane, B žádá R a čeká
- A žádá S a dostane, A uvolní R a S
- B čekal na R a dostane, B žádá S a dostane