PB153
Operační systémy a jejich rozhraní

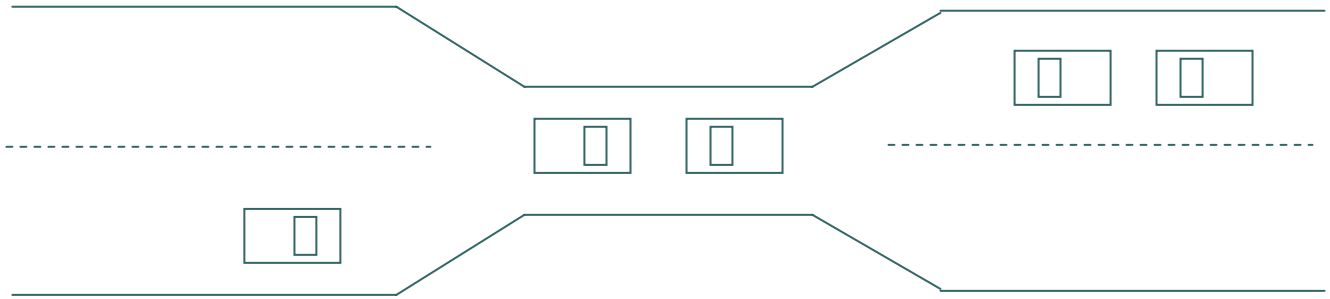
Uváznutí

Problém uváznutí

- Existuje množina blokovaných procesů, každý proces vlastní nějaký prostředek (zdroj) a čeká na zdroj držený jiným procesem z této množiny
- Příklad 1
  - v systému existují 2 páskové mechaniky
  - procesy P_1 a P_2 chtějí kopírovat data z pásky na pásku, každý z procesů „vlastní“ jednu mechaniku a požaduje alokací druhé
- Příklad 2
  - Semafory A a B, inicializované na 1

P_0	P_1
wait (A);	*wait(B)*
wait (B);	*wait(A)*

Příklad: úzký most



- Most s jednosměrným provozem
- Každý vjezd mostu lze chápat jako zdroj
- Dojde-li k uváznutí, lze ho řešit tím, že se jedno auto vrátí
  - Preempce zdroje (přivlastnění si zdroje, který vlastnil někdo jiný) a vrácení soupeře do situace před žádostí o přidělení zdroje (preemption a rollback)
- Při řešení uváznutí se může vracet i více vozů
- Může docházet ke stárnutí

Definice uváznutí a stárnutí

- Uváznutí
  - množina procesů *P* uvázla, jestliže každý proces P_i z *P* čeká na událost (uvolnění prostředku, zaslání zprávy), kterou vyvolá pouze některý z procesů *P*
- Stárnutí
  - požadavky 1 nebo více procesů z *P* nebudou splněny v konečném čase
    - z důvodů vyšších priorit jiného procesu
    - z důvodů prevence uváznutí apod.

Model

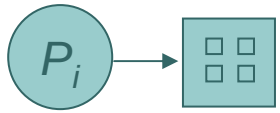
- Typy zdrojů R_1, R_2, …, R_m
  - tiskárna, paměť, I/O zařízení, …
- Každý zdroj R_i má W_i instancí
- Každý proces používá zdroj následujícím způsobem
  1. žádost
  2. použití
  3. uvolnění (v konečném čase)

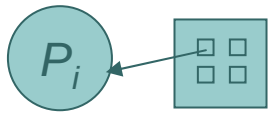
Charakteristika uváznutí

- K uváznutí dojde, když začnou současně platit 4 následující podmínky
  - vzájemné vyloučení (mutual exclusion)
    - sdílený zdroj může v jednom okamžiku používat pouze jeden proces
  - ponechání si zdroje a čekání na další (hold and wait)
    - proces vlastnící alespoň zdroj čeká na získání dalšího zdroje, dosud vlastněného jiným procesem
  - bez předbíhání (no preemption)
    - zdroj lze uvolnit pouze procesem, který ho vlastní, dobrovolně po té, co daný proces zdroj dále nepotřebuje
  - kruhové čekání (circular wait)
    - existuje takový seznam čekajících procesů (P_0, P_1, …, P_n), že P_0 čeká na uvolnění zdroje drženého P_1, P_1 čeká na uvolnění zdroje drženého P_2, …, P_{n-1}čeká na uvolnění zdroje drženého P_n, a P_n čeká na uvolnění zdroje drženého P_0

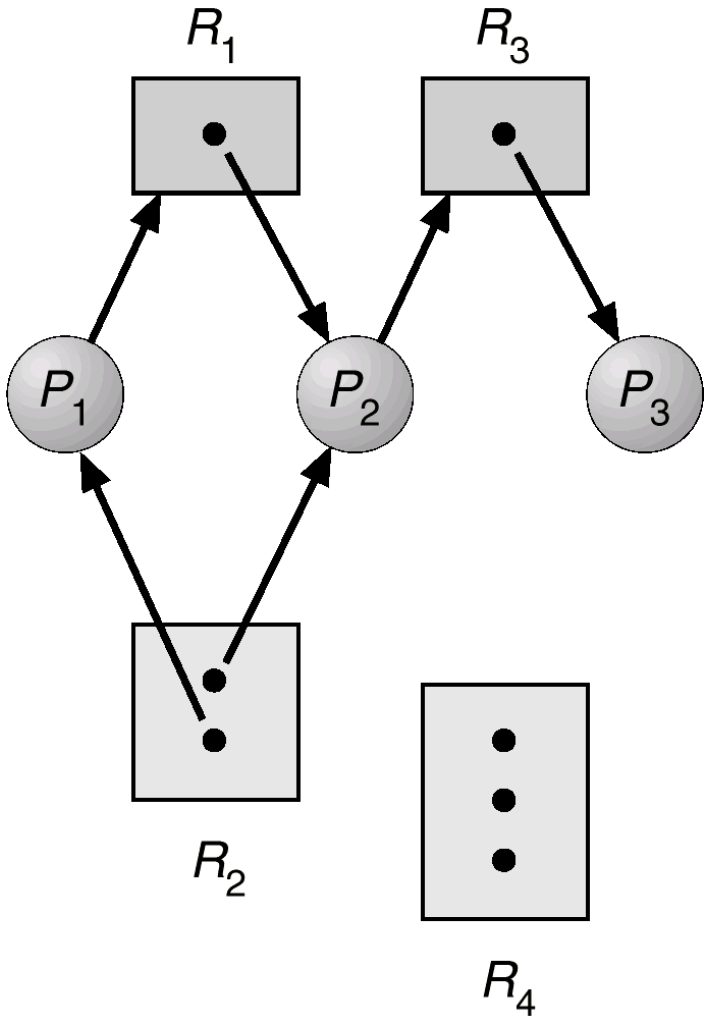
Graf přidělení zdrojů

- Resource-Allocation Graph, RAG
- Množina uzlů V a množina hran E
- uzly jsou dvou typů:
  - $P = \{P_1, P_2, \ldots, P_n\}$, množina procesů existujících v systému
  - $R = \{R_1, R_2, \ldots, R_m\}$, množina zdrojů existujících v systému
- Hrana požadavku – orientovaná hrana $P_i \rightarrow R_j$
- Hrana přidělení – orientovaná hrana $R_j \rightarrow P_i$
- Proces: 

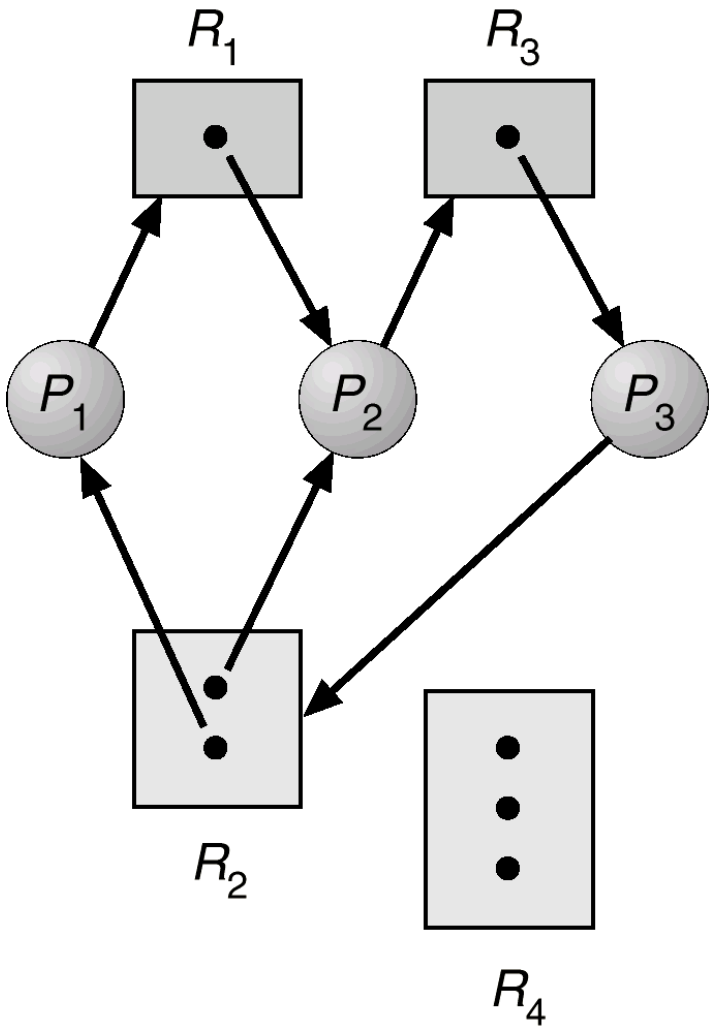
- Zdroj se 4 instancemi: 
- Proces P_i požadující prostředek R_j: 

- Proces P_i vlastnící prostředek R_j : 

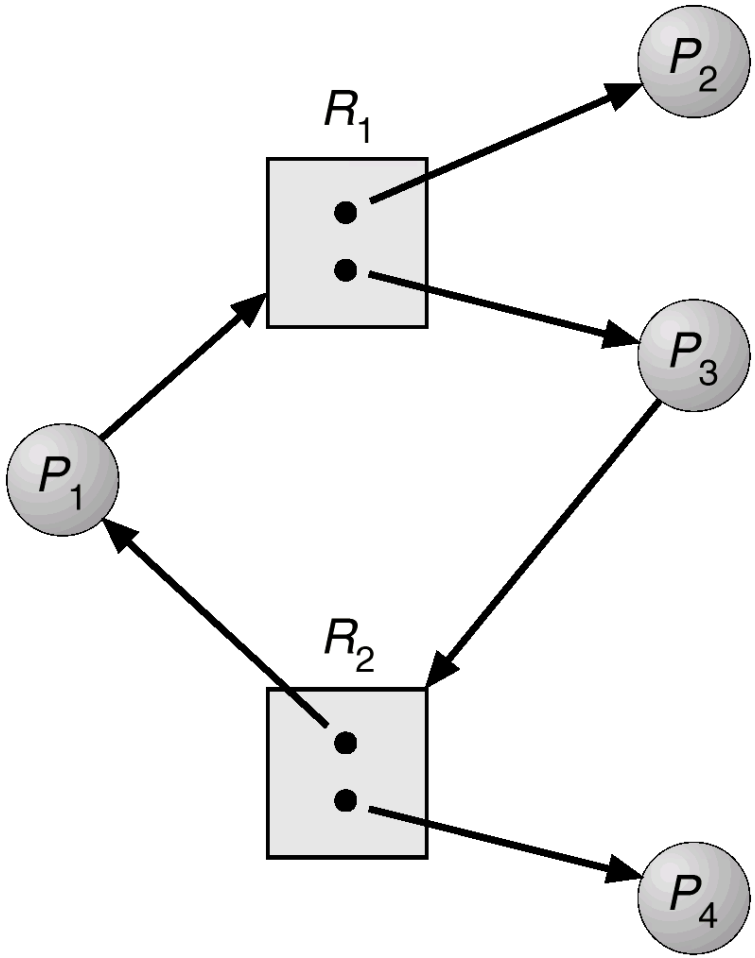
Příklad RAG (bez cyklu)



Příklad RAG (s uváznutím)



Příklad RAG (bez uváznutí)



RAG: závěry

- Jestliže se v RAG nevyskytuje cyklus – k uváznutí nedošlo
- Jestliže se v RAG vyskytuje cyklus
  - existuje pouze jedna instance zdroje daného typu
    → k uváznutí došlo
  - existuje více instancí zdroje daného typu
    → k uváznutí může (ale nemusí) dojít

Problém uváznutí

- Ochrana před uváznutím prevencí
  - zajistíme, že se systém nikdy nedostane do stavu uváznutí
  - zrušíme platnost některé nutné podmínky
- Obcházení uváznutí
  - detekce potenciální možnosti vzniku uváznutí a nepřipuštění takového stavu
  - zamezujeme současné platnosti všech nutných podmínek
  - prostředek se nepřidělí, pokud by hrozilo uváznutí (hrozí stárnutí)
- Obnova po uváznutí
  - uváznutí povolíme, ale jeho vznik detekujeme a řešíme
- Ignorování hrozby uváznutí
  - uváznutí je věc aplikace ne systému
  - způsob řešení zvolený většinou OS

Ochrana prevencí

- Nepřímé metody
  - zneplatnění některé nutné podmínky
    - Virtualizací prostředků, ruším nutnost vzájemné výlučnosti při přístupu
    - požadováním všech prostředků najednou
    - odebíráním prostředků
- Přímé metody
  - nepřipuštění platnosti postačující podmínky (cyklus v grafu)
    - uspořádání pořadí vyžadování prostředků

Prevence uváznutí (1)

- Vzájemné vyloučení
  - podmínka není nutná pro sdílené zdroje
  - u nesdílených zdrojů musí podmínka platit
  - řeší se např. virtualizací prostředků (např. tiskárny)
- Ponechání zdrojů a čekání na další
  - při žádosti o zdroje proces žádné zdroje „vlastnit“ nesmí
  - proces musí požádat o zdroje a obdržet je dříve než je spuštěn běh procesu
  - důsledkem je nízká efektivita využití zdrojů a možnost stárnutí

Prevence uváznutí (2)

- Zakázané předbíhání
  - jestliže proces držící nějaké zdroje a požadující přidělení dalšího zdroje, nemůže zdroje získat okamžitě, pak se uvolní všechny tímto procesem držené zdroje
  - „odebrané“ zdroje se zapíší do seznamu zdrojů, na které proces čeká
  - proces bude obnoven, pouze jakmile může získat jak jím původně držené zdroje, tak jím nově požadované zdroje
- Zabránění kruhovému pořadí
  - zavedeme úplné uspořádání typů zdrojů a každý proces bude žádat o prostředky v pořadí daném vzrůstajícím pořadí výčtu

Obcházení uváznutí

- Systém musí mít nějaké dodatečné apriorní informace
- Nejjednodušší a nejužitečnější model požaduje, aby každý proces udal maxima počtu prostředků každého typu, které může požadovat
- Algoritmus řešící obcházení uváznutí dynamicky zkouší, zda stav systému přidělování zdrojů zaručuje, že se procesy v žádném případě nedostanou do cyklické fronty čekání
- Stav systému přidělování zdrojů se definuje počtem dostupných a přidělených zdrojů a maximem žádostí procesů

Detekce uváznutí

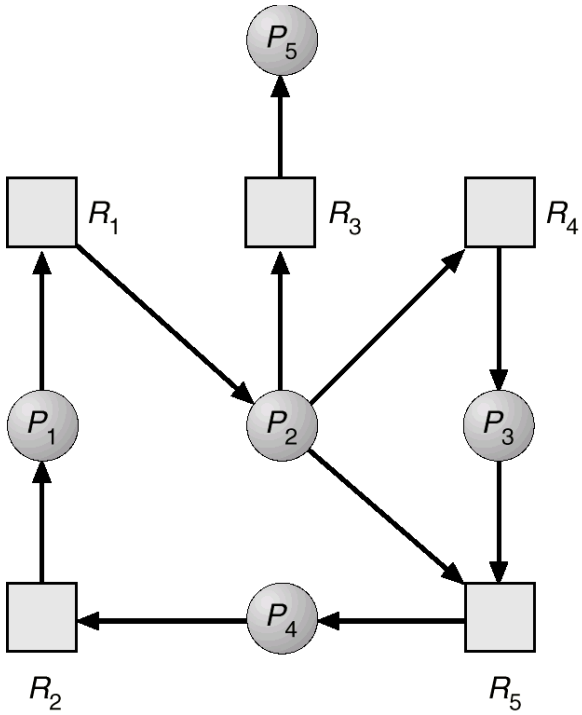
- Umožníme, aby došlo k uváznutí
- Ale toto uváznutí detekujeme
- Aplikujeme plán obnovy

1 instance prostředku každého typu

- Udržuje se graf čekání (wait-for graph)
  - uzly jsou procesy
  - $P_i \rightarrow P_j$ jestliže P_i čeká na P_j
- Periodicky se provádí algoritmus, který v grafu hledá cykly
- Algoritmus pro detekci cyklu v grafu požaduje provedení n^2 operací, kde n je počet uzlů v grafu

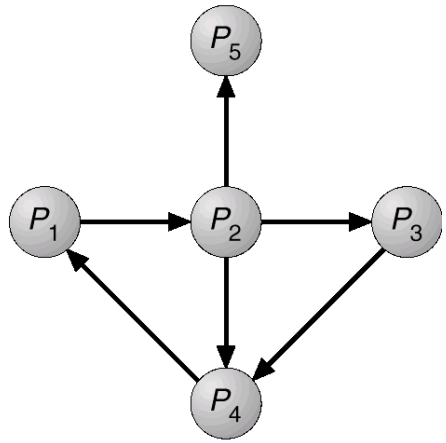
Grafy

Graf přidělení zdrojů



(a)

Odpovídající graf čekání



(b)

Obnova: ukončení procesu

- Násilné ukončení uváznutých procesů
- Násilně se ukončuje jednotlivě proces po procesu, dokud se neodstraní cyklus
- Čím je dáno pořadí násilného ukončení?
  - priorita procesu
  - doba běhu procesu, doba potřebná k ukončení procesu
  - prostředky, které proces použil
  - prostředky, které proces potřebuje k ukončení
  - počet procesů, které bude potřeba ukončit
  - preference interaktivních nebo dávkových procesů

Obnova: nové rozdělení prostředků

- Výběr oběti: minimalizace ceny
- Návrat zpět (rollback) – návrat do některého bezpečného stavu, proces restartujeme z tohoto stavu
- Stárnutí – některý proces může být vybírán jako oběť trvale
  - řešení: do cenové funkce zahrneme počet restartů (rollbacků)