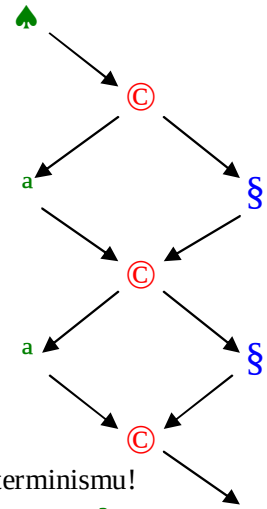
2003-12-12

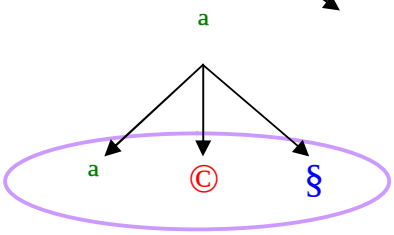
Příklad – kopírování dat:

♠ čti blok dat do S
♣ zapiš blok dat z T
♥ přesuň v paměti S → T



Deterministické zpracování

Současné spuštění všech tří procesů vede k nedeterminismu!



Nedeterminismus

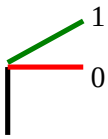
Pokud ♣ začne až po ukončení ♥, ale před druhým ♥ a ♠ až po ♥ bude prvý záznam překopírován správně. Začne-li ♣ dříve, překopíruje se nesmysl (původní obsah paměti). Při jiném průběhu se záznam může ztratit (být přepsán dříve než zapsán) nebo být zapsán několikrát**.**
Nedeterminismus byl způsoben tím, že procesy užívají společné proměnné S a T a přístup k nim není řízen.

Kritická oblast (sekce) = část kódu programu, která pracuje se sdílenými daty a kterou je potřeba vymezit v každém paralelním procesu a přístup do ní řídit (synchronizovat)
Zásady:

1. V kritické oblasti může být současně nejvýše jeden proces. Je-li kritická oblast obsazena, musí být proces převeden do stavu „čeká na prostředek“, do doby než oblast stávající proces opustí.
2. Každý proces, který vstoupí do kritické oblasti ji musí v konečném čase opustit.
3. Každý proces, který čeká na vstup do kritické oblasti se v konečném čase musí vstupu do ní dočkat (nesmí dojít k vyhladovění procesu).

1. ∧ 2. ⇒ 3.

Prostředky pro řízení přístupu do kritické oblasti (synchronizační primitiva)
Zámek (binární semafor) = sdílená proměnná s možnými polohami:
1 = otevřen
0 = uzavřen

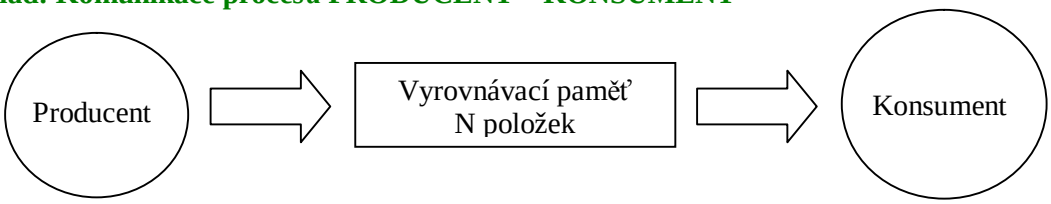


- Proces před vstupem do kritické oblasti testuje polohu semaforu. Je-li otevřen, pokračuje a semafor ihned uzavře. Je-li semafor zavřen, přejde do stavu čeká.
- Po opuštění kritické oblasti proces semafor otevře.
- Obsluha semaforu je spojena s převodem procesů ze stavu do stavu. Musí ji tedy provádět operační systém.
- Semafor sám je sdílenou proměnnou. Jeho obsluha tedy tvoří také kritickou sekci.
- Proto nelze test semaforu provádět ve více instrukcích (mezi nimi by mohlo dojít k přerušení) ⇒ potřeba speciální instrukce TST (TSTL) – „test and set (lock)“. Jedna instrukce realizuje kritickou sekci – je nepřerušitelná. Přerušení se uplatní až po ukončení instrukce. Je realizováno HW.

Operace nad binárním semaforem S:
wait(S) – je-li S otevřen, **ihned** jej zavře a pokračuje, je-li zavřen, zavolá operační systém a ten zařadí proces do stavu PŘIPRAVEN
signal(S) –otevře semafor a volá operační systém, který rozhodne o zařazení vhodného procesu (podle zvolené disciplíny obsluhy)

Někdy je vhodný počítaný semafor (typu INTEGER nebo BYTE).
Například pro problém „producent – konsument“ s omezeným rozsahem vyrovnávací paměti.
S = 1 2, ..., N otevřen
S = 0 zavřen
wait (S): (při vstupu):
Je-li S > 0, pak **ihned** S –> S – 1 a pokračuje se,
Je-li S = 0, volá se operační systém, který proces odstaví
signal(S): (při opuštění) S = S + 1a pokud nějaký proces čeká, bude zařazen.
Realizovat lze pomocí binárních semaforů.

Příklad: Komunikace procesů PRODUCENT – KONSUMENT



Semafory: PRAZDNYCH na začátku N
PLNYCH na začátku 0
Lze realizovat i pomocí počtu vyslaných a počtu přijatých signálů a hlídání rozdílu
0 < PocetVyslanzch – PocetPrijatych < N

Příklad: „čtenáři – údržbáři“ datové základny (čtenáři se vzájemně nevylučují. Údržbář se vylučuje se čtenáři i s jinými údržbáři)

Přidělování zdrojů (prostředků):

Proces potřebuje zdroje (= prostředky) → HW (I/O zařízení, paměť)
→ SW (data, programy)

Klasifikace zdrojů (z hlediska multitaskového režimu):
- Násobně použitelné (= sdílitelné, = reentrantní)
  Lze je přidělit jinému procesu před tím, než je proces, kterému byly přiděleny dokončen. Původní proces může být ve stavu „čeká“ nebo „připraven“. Přidělí je krátkodobý plánovač – dispečer OS.
  (základní jednotka, disk jako celek, myš, data pro čtení)
- Opakovaně použitelné (= reusabilní).
  Lze je přidělit jinému procesu až poté, co proces, kterému byly přiděleny dosud byl ukončen. Přiděluje je dlouhodobý plánovač – řízení úloh.
  (tiskárna, pracovní oblast v paměti a na disku, většina programů, pokud obsahují pracovní položky)
- Na jedno použití (= nereusabilní)
  Nelze přidělit znova.
  (papír do tiskárny, ale také třeba špatně vytvořený program, který užívá stav paměti po překladu a před prvým výpočtem – opomenuté nulování střadače)

Práce s reusabilním, ale ne reentrantním zdrojem tvoří kritickou oblast. Nutno řídit přístup k ní.
Přidělování opětovně použitelných, avšak nesdílitelných prostředků vyžaduje zvláštní pozornost. Je třeba zabránit uváznutí (dead lock).

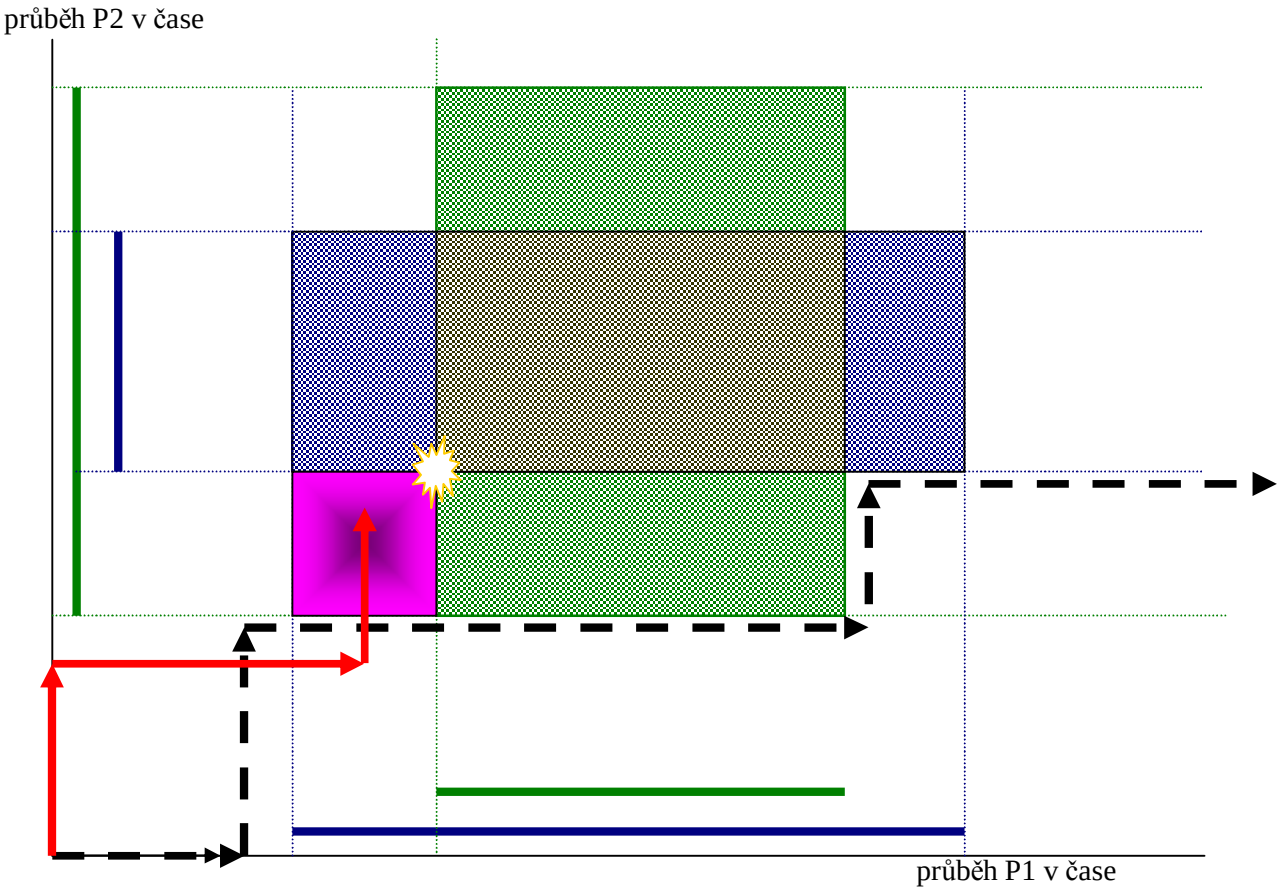
UVÁZNUTÍ
= zamrznutí, zaseknutí, smrtelné objetí, smrtelný zámek, dead lock
= stav, kdy všechny spuštěné procesy jsou ve stavu „čeká“ – na událost, která nenastane)

Příklad: Proces P1 potřebuje zdroj Z1 a poté Z2, proces P2 potřebuje Z2 a poté Z1

Nebezpečná oblast. Vstup do ní ⇒ budoucí uváznutí.

„Úspěšný“ průběh: P1 obsadí Z1, obsadí Z2, P2 na uvolnění Z2 počká, poté uvolní Z2, P2 obsadí Z2 a počká až P1 Z1 uvolní ...

Zpracování nutně uvázne. P1 obsadí Z1 a drží jej. P2 obsadí Z2 a drží jej. P1 čeká na Z2, P2 čeká na Z1 ⇒ systém vždy uvázne.



Jak zabránit uváznutí?

1. Následné řešení : DETEKCE A ZOTAVENÍ
   V určitých časových intervalech je (na základě přerušení od hodin) spouštěn systémový program („démon“), který kontroluje, zda nejsou všechny spuštěné uživatelské procesy ve stavu „čeká“. Pokud ano, některý zruší (nebo vyzve obsluhu ke zrušení).
   Nevýhoda: Důsledek je porucha zpracování. Zrušený proces neproběhne.
2. PREVENCE (vždy lepší než následná terapie)
   A) **Úplné přidělování:** Procesu jsou přiděleny všechny reusabilní prostředky ihned po spuštění, i když je bude potřebovat až později.
   HODNOCENÍ: Plýtvá se prostředky.
   B) **Uspořádané přidělování:** Zdroje seřazeny. Současně s přidělením prostředku se přidělí i všechny, které jsou v tomto pořadí před ním.
   HODNOCENÍ: Rovněž plýtvání, i když někdy o něco (málo) menší.
   C) **Řízené přidělování:** Algoritmus bankéře.
   Při každé žádosti o přidělení prostředku se zkoumá nejen, zda je tento prostředek k dispozici, ale také, zda existuje další bezpečná strategie dalšího přidělování, která umožní všechny procesy dokončit.
   Název algoritmu podle analogické ekonomické úlohy:
   Bankéř má základní kapitál a několik zákazníků. Jeho kapitál postačí pokrýt potřeby každého ze zákazníků jednotlivě, ne však všech současně. Zákazníci žádají o uvolnění prostředků na své investice postupně. Půjčené prostředky vrátí až po dokončení celé investice.
   Příklad:
   Základní kapitál banky 20 mil. 3 zákazníci. Každý potřebuje na celou investici 10 mil. Průběh:
   1. Prvý žádá na stavbu 6 mil. LZE VYHOVĚT. Bance zbude 14 mil.
   2. Druhý žádá na stavbu též 6 mil. LZE VYHOVĚT. Bance zbude 8 mil.
   3. Třetí žádá rovněž 6 mil.Kdyby je banka půjčila, zbyly by jí jen 2 mil. a to nestačí k dokončení žádné s investic, které je nutné pro zahájení splátek. PŮJČIT LZE JEN 2 mil., aby zůstalo 6 k dokončení jedné z investic.

   HODNOCENÍ: Dobré využití zdrojů. Poměrně náročný algoritmus zdržuje.