Interakce mezi procesy a typické problémy paralelismu

Z FITwiki

Obsah

Interakce mezi procesy

Interakce mezi procesy

- **kooperace** (spolupráce na řešení úkolu, je potřeba synchronizace)
- **soupeření** (o omezené zdroje, je potřeba výlučný přístup)

Podrobněji v: Synchronizace, vzájemné vyloučení

Řešení vzájemného vyloučení

Hardwarové

Podrobněji v: Implementace vzájemného vyloučení

V operačním systému

Podrobněji v: Semafory, Monitory

Softwarové

- předpokládáme
  - čtení/zápis jedné hodnoty je atomické (nemusí platit při používání stránkování)
  - současné čtení/zápis budou provedeny v náhodném pořadí za sebou

Dekkrův algorimus (Dekker's algorithm)

```
shared var flag: array [0..1] of boolean; //flagy žádosti o vstup do kritické sekce
turn: 0..1; // označení aktuálního kola

repeat
  flag[i] := true;            // proces žádá o vstup do kritické sekce (nastaví flag žádosti)
    while flag[j] do          // proces ověřuje dokud o vstup žádá i druhý proces
      if turn=j then          // Pokud je právě kolo druhého procesu ...
        flag[i] := false;     // ... proces se vzdá své žádosti o krit. sekci...
        while turn=j do skip; // ... a čeká dokud neskončí kolo druhého procesu
        flag[i] := true;      // Poté znovu nastaví svoji žádost o krit. sekci
      endif
    endwhile                  // Proces vstupuje do krit. sekce pokud druhý proces flag nenastavil
  <critical section>
  turn := j;                  // Proces nastaví kolo na druhý proces
  flag[i] := false;
  <remainder section>
until false;
```

Petersonův algoritmus (Peterson's algorithm)

```
shared var flag: array [0..1] of boolean; //flagy žádosti o vstup do kritické sekce
turn: 0..1; // označení aktuálního kola

repeat
  flag[i] := true;                      // Proces žádá o krit. sekci
  turn := j;                            // Proces nastaví kolo na druhý proces
  while (flag[j] and turn=j) do skip;   // Čeká dokud je kolo druhého procesu a žádá o krit. sekci
  <critical section>
  flag[i] := false;                     // Po dokončení krit. sekce proces stáhne žádost o krit se
  <remainder section>
until false;
```

Operátor `<await B->S>`

- je původně pouze teoretický operátor, implementovaný ve vyšších programovacích jazycích pro paralelní programování.

- implementace je problémová
- Zajišťuje atomičnost `<...>`, očekává splnění podmínky `B` a poté atomicky vykoná sekvenci příkazů `S`.

Critical Region

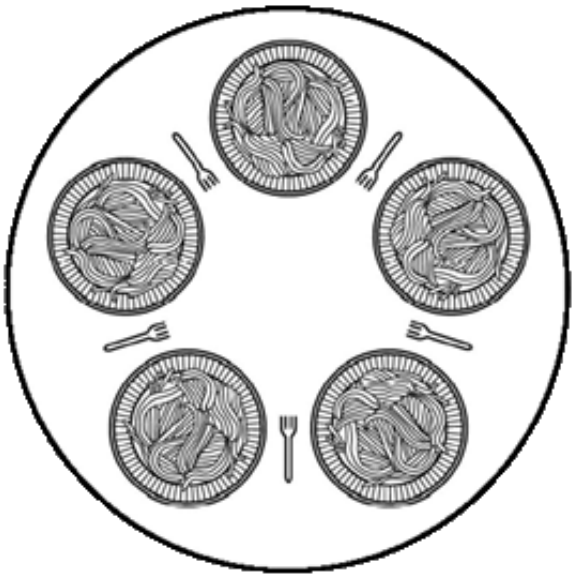
- ve vyšších jazycích
- jsou obalení použití semaforů pro přístup ke KS
- klíčové slovo `region` (označení KS) a označení proměnných `shared` (všechny přístupy k proměnné vzájemně vyloučené)
- Problém, pokud se zanořují, pak může dojít ke konfliktu.
- Jednoduchá implementace (téměř makro).

Conditional Critical Region

- rozšiřuje koncepci o podmínku
- pokud je stanovena dobře, ke kolizi nedojde
- Implementace je ovšem velmi složitá.

Typické problémy paralelismu

Večeřící filozofové (Dining philosophers)



- 5 filozofů
- 5 vidliček
- filozofové přemýšlejí, pak dostanou hlad, nají se a opět přemýšlejí
- Každý filozof potřebuje 2 vidličky aby se najedl
- Problém: vyhladovění, deadlock

Čtenáři a písaři (Readers/writers)

- sdílená proměnná

- několik čtenářů, kteří chtějí číst hodnotu
- jeden nebo několik písařů měnících hodnotu
- číst může libovolný počet zaráz
- pokud se zapisuje nikdo jiný nesmí zapisovat ani číst

Producenti a konzumenti (producer/consumer problem)

- producenti produkují data
- konzumenti produkovaná data přijímají a spotřebovávají
- vyžaduje buffer pro ukládání dat
- nutné zaručit, že se nebude číst prázdný buffer
- pokud má buffer omezenou velikost je nutná zabránit přetečení
- vzájemné vyloučení v přístupu k bufferu

Kategorie: Státnice 2011 | Paralelní a distribuované algoritmy

---

- Stránka byla naposledy editována 26. 5. 2011 v 08:38.