Monitory

Z FITwiki

Monitor

> je abstraktní datový typ, který zajíistí vzájemné vyloučení operací nad monitorem = všechny operace monitoru jsou atomické

- Monitor se skládá z dat, ke kterým je potřeba řídit přístup, a množiny funkcí, které nad těmito daty operují.
- Sdílené proměnné dostupné pouze operacemi monitoru.
- Odděluje čekání a operace se sdílenými proměnnými
- Provádění operací jednoho monitoru je vzájemně vyloučené.
- Inicializace monitoru nastaví sdílené proměnné.
- Používá se namísto obecných semaforů v UNIXu (vyloučení i signály).
- Snazší používání (vyšší úroveň abstrakce), z pohledu programátora transparentní (jen volá funkce, zamykání řeší monitor)

- Proces pokud chce do monitoru vstoupit (zavolat jeho funkci) musí nejdříve získat zámek, pokud má někdo jiný zámek, proces se zablokuje a čeká
- nelze čekat aktivně uvnitř monitoru
  - uvnitř monitoru condition c - fronta čekajících procesů
  - operace:
    - c.wait - pozastavení procesu, vzdání se monitoru
    - c.signal - odblokování prvního čekajícího (pokud je), ten získá opět výlučný přístup k monitoru
  - problém: operace se sdílenou proměnnou musí být uvnitř monitoru → po c.signal jsou dva procesy v monitoru
  - řešení: někdo musí být pozastaven - blokující/neblokující signalizace podmínky
- 2 typy:
  - Hoare, Hansen - proces P provádějící c.signal je pozastaven a pokračuje odblokovaný proces Q za c.wait. Až Q opustí monitor nebo začne zase čekat, pokračuje nejprve P a pak teprve ostatní procesy čekající na vstup do monitoru. Signalizace stavu je kooperativní, ten kdo stav změnil, efektivně předá monitor probuzenému čekajícímu procesu. Pokud nikdo nečeká, je signalizace prázdnou operací.
  - Lampson, Redell (Mesa, 1980) - operace c.notify pouze odblokuje pozastavený proces, monitor zůstane dále stejnému procesu, teprve po opuštění monitoru se může odblokovaný proces dostat do monitoru a pokračovat za c.wait, soutěží ovšem s procesy vstupujícími do monitoru normálně - nelze zaručit splnění testované podmínky v odblokovaném procesu! Proto je třeba vždy testovat podmínku čekání po probuzení znovu!

Podmíněné proměnné

> slouží k čekání uvnitř monitoru (proces v monitoru čeká na splnění nějaké podmínky)

Když funkce monitoru potřebuje počkat na splnění podmínky, vyvolá operaci wait na podmíněné proměnné, která je s touto podmínkou svázána. Tato operace proces zablokuje, zámek držený tímto procesem je uvolněn a proces je odstraněn ze seznamu běžících procesů a čeká, dokud není podmínka splněna. Jiné procesy zatím mohou vstoupit do monitoru (zámek byl uvolněn). Pokud je jiným procesem podmínka splněna, může funkce monitoru „signalizovat“, tj. probudit čekající proces pomocí operace notify. Operace notify budí jen ty procesy, které provedly wait na stejné proměnné.

POSIX

Proměná typu `p_thread_cond_t` ve spojení se vzájemným vyloučením = monitor

```
pthread_mutex_t mutex=PTHREAD_MUTEX_INITIALIZER;
pthread_cond_t cond = PTHREAD_COND_INITIALIZER;
```

```
int suma = 0;           /* testovaná podmínka */
...
pthread_mutex_lock(&mutex);
while (suma == 0) { /* podmínka splněna? */
  pthread_cond_wait(&cond, &mutex);/*ne, čekat*/
}
...     /* zpracuj suma */
pthread_mutex_unlock(&mutex);
```

Signalizace podmínky

```
int pthread_cond_signal(pthread_cond_t *cond);
int pthread_cond_broadcast(pthread_cond_t *con);
```

Pokud čeká proces na cond, je odblokován (případně všechny čekající pro broadcast)

Použití signalizace

```
pthread_mutex_lock(&mutex);
suma = suma + neco;         /* změna stavu podmínky */
pthread_cond_signal(&cond);
pthread_mutex_unlock(&mutex);
```

Citováno z „http://wiki.fituska.eu/index.php?title=Monitory&oldid=9867“
Kategorie: Státnice 2011 | Pokročilé operační systémy

- Stránka byla naposledy editována 6. 6. 2012 v 17:06.