Meziúlohové synchronizační mechanismy, Kritická sekce)

Důležité pojmy:

Atomicita neboli nedělitelnost je důležitou vlastností v **programování**. Znamená, že daná činnost (operace) se provede najednou, nemůže být přerušena něčím jiným a později dokončena.

Kritická sekce

- Část programu, kde procesy používají sdílené prostředky (např. sdílená paměť, sdílená proměnná, sdílený soubor, …).
- Sdružené kritické sekce
  - Kritické sekce dvou (nebo více) procesů, které se týkají stejného sdíleného prostředku.
- Vzájemné vyloučení
  - Procesům není dovoleno sdílet stejný prostředek ve stejném čase.
  - Procesy se nesmí nacházet ve sdružených sekcích současně.

Pro synchronizaci přístupu úloh do kritické sekce (synchronizaci paralelních procesů) se používají meziúlohové synchronizační mechanismy (synchronizační primitiva).
Mezi ně patří:

- Semafor (Semaphore)
- Spinlock
- Lock

Synchronizační primitiva jsou v operačních systémech prostředky, umožňující zároveň běžícím aplikacím ošetřit současný přístup ke sdíleným prostředkům. Ve smyslu algoritmu se jedná o rozhraní a jeho implementace není důležitá.

Chybné použití synchronizačních primitiv může vést k jejich neúčinnosti (tedy k prostředku stejně mohou přistoupit dva procesy najednou) nebo k deadlocku (vzájemnému zablokování).

Mezi synchronizační primitiva patří zámek (lock) a jejich zobecnění semafory, fronty zpráv a monitor.

Zámky a semafory bývají implementovány operačním systémem pomocí atomických operací na sdílené paměti a plánovače. Pro synchronizaci v paralelním programování stačí atomické operace na sdílené paměti (čekají na sebe procesy na různých procesorech a tedy mohou čekat aktivně) a je možné je implementovat i bez pomoci operačního systému.

Fronty zpráv jsou primitivní operací v případě paralelního programování, ale je možné je implementovat v operačním systému i na jednom procesoru.

Monitor je možné realizovat pouze s podporou programovacího jazyka.

Pro uvedená synchronizační primitiva platí, že jsou vzájemně ekvivalentní - tedy že pokud máte k dispozici jedno z nich, můžete s jeho pomocí naimplementovat ty ostatní. S výjimkou zámků to lze navíc dokázat bez aktivního čekání: na implementaci pomocí zámků potřebujete spolupráci scheduleru (možnost uspat program a později ho probudit) nebo aktivně čekat (Stačí jeden „zámkový server“ neustále prohlížející oblasti sdílené paměti a interpretující je jako zprávy. Ostatní procesy pak mohou čekat na zámcích, které vlastní a uvolní právě tehdy, když jim vyřizuje zprávu.) Pro implementaci semaforů pomocí front zpráv procesu potřebujete „semaforový server“ starající se o semafor, ale ten čeká pasivně. Pro fronty pojmenované server nepotřebujete.

Lock (zámek)

Zámek je synchronizační primitivum, které má dvě funkce. *Lock()* a *unlock().* (Zamkni, odemkni) a tomu odpovídající stavy: zamknutý/odemknutý zámek. Na začátku kritické sekce se zavolá lock, po jejím opuštění unlock. Pokud proces chce zamknout již zamčený zámek, zařadí se do fronty čekajících procesů na tomto zámku a proces se uspí. Toto provede obsluha rutiny lock(). V momentě, kdy proces odemyká zámek a fronta čekajících procesů je prázdná, obsluha rutiny unlock() změní stav zámku na odemknuto. Pokud je fronta neprázdná, vybere jeden z čekajících procesů, který se pustí do kritické sekce. (Přesněji řečeno, bude přeřazen ze stavu spí do stavu připraven, a běžet začne v momentě, kdy bude naplánován, viz obr. 1).

Semafor

Semafor je synchronizační primitivum obsahující celočíselný čítač, který si lze představit například jako počítadlo volných prostředků. Poskytuje atomické operace „up“ a „down“. Operace „down“ sníží čítač o jedničku, v případě, že už je nulový (nedostává se prostředků), se proces zablokuje a přidá do fronty procesů čekajících na daný semafor. Operace „up“ zkontroluje frontu, a v případě, že je neprázdná, vybere jeden proces čekající ve frontě a odblokuje jej (ten pak pokračuje za svou operací „down“); je-li fronta prázdná, zvýší hodnotu čítače o jedničku.

Spinlock

Spinlock je v operačních systémech druh zámku, na nějž je třeba aktivně čekat – čekající proces tedy při čekání na spinlock spotřebovává systémové prostředky.

Spinlocky se zpravidla používají pouze v operačním systému, aplikacím jsou poskytována složitější synchronizační primitiva, které čekající aplikace uspí a zařadí do fronty, takže v době, kdy jsou zablokovány, může běžet něco jiného. Na druhou stranu, tyto složitější struktury vyžadují ochranu svých dat proti vícenásobného přístupu, a k tomu lze použít právě jednodušší a rychlejší spinlocky.