INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

VYSOKÁ ŠKOLA BÁŇSKÁ – TECHNICKÁ UNIVERZITA OSTRAVA
FAKULTA STROJNÍ

PROGRAMOVÁNÍ ŘÍDÍCÍCH SYSTÉMŮ

Komunikace a synchronizace procesů

Ing. Ivo Špička, Ph.D.

Ostrava 2013

ISBN 978-80-248-3041-4

Tento studijní materiál vznikl za finanční podpory Evropského sociálního fondu (ESF) a rozpočtu České republiky v rámci řešení projektu: CZ.1.07/2.2.00/15.0463, MODERNIZACE VÝUKOVÝCH MATERIÁLŮ A DIDAKTICKÝCH METOD

OBSAH

11 KOMUNIKACE A SYNCHRONIZACE PROCESŮ

Obsah kapitoly:

Komunikace a synchronizace procesů

CÍL:

Po prostudování tohoto odstavce budete umět:

- definovat zasílaní zpráv, monitory, semafor, signál, kritické sekce vzájemného vyloučení, souběh, uváznutí a stárnutí
- popsat realizaci některých synchronizačních nástrojů, podmíněný a nepodmíněný zákaz přepínání kontextu

11.1 ZASÍLÁNÍ ZPRÁV

Technika zasíláni zpráv představuje poměrné jednoduchou abstrakci komunikace procesů spojenou s jejich synchronizací. Komunikace pomocí zasílání zpráv probíhá tak, že jeden proces vyšle jednou operací posloupnost bitů konečné délky (zašle zprávu) a druhý proces ji přijme. Potřebné synchronizační operace jsou tedy typicky dvě procedury

SEND MESSAGE(...) pro zaslání zprávy a
RECEIVE MESSAGE(...) pro přijetí zprávy.

Konkrétní operace se liší dle toho, jak je určen druhý proces s ním se komunikuje - tzv. adresací a dále tím, probíhá-li komunikace synchronně nebo asynchronně.
Adresace se nazývá symetrická, uvádí-li se při komunikaci jméno vysílajícího i přijímajících asymetrická, uvádí-li se jen jméno přijímajícího procesu. Jméno vysílajícího procesu se určí při přijetí zprávy dotazem přes parametr. Adresace se nazývá nepřímá, uvádí-li se jméno použité vyrovnávací paměti nebo komunikačního kanálu.
Asynchronní vysílání vyžaduje vyrovnávací paměť o určité kapacitě. Technika výběru se může realizovat bud' prostřednictvím schránek paměti, staticky, přičemž jsme omezeni předem zvolenou kapacitou vyrovnávací paměti. Rafinovaněji způsob představuje technika výběru pomocí fronty, dynamická, kde je kapacita vyrovnávací paměti omezena pouze kapacitou operační paměti dostupnou pro dynamické přidělování paměti.
Synchronní operace RECEIVE obsahuje čekání na příchod zprávy a operace SEND čekání na příchod potvrzení o přijetí vyslané zprávy, resp. potvrzení zahájení komunikace. Při tomto režimu komunikace stačí jedna vyrovnávací paměť s kapacitou maximálně jedné zprávy.

11.2 MONITORY

Základní myšlenkou monitoru je spojit deklaraci společné proměnné - resp. společné datové struktury - s explicitní definicí všech operací, které lze na ní provádět a současně stanovit, že tyto operace jsou zároveň vyloučeny. Koncepce monitoru vychází a myšlenky abstraktního datového typu a rozšiřuje ji o vzájemné vyloučení operací.
Při implementaci je třeba ještě dodat 2 operace realizované na úrovni operačního systému

vstup_do_monitoru(monitor M);
výstup_z monitoru(monitor M);

pomocí nichž synchronizujeme operace p1, p2,...,pn v monitoru tak, aby probíhaly ve vzájemném vyloučení. To je však velmi silná podmínka protože často stačí jestliže například pomocí zdroje synchronizujeme je některé operace.

11.3 REALIZACE NĚKTERÝCH SYNCHRONIZAČNÍCH NÁSTROJŮ.

Již jsme uvedli, že pro synchronizaci procesů se využívají tyto dva základní principy:

A. Aktivní čekání, kdy dochází k tomu, že do procesu vkládáme pomocné prázdné akce, čímž dojde ke "vzdálení" zakázané oblasti tvořené průnikem kritických oblastí.
B. Pasivní čekání, při němž dochází k pozastavení všech konkurujících virtuálních procesorů a tím k pozastavení jim odpovídajících procesů.

11.3.1 Aktivní čekání

Bylo již řečeno, že technika aktivního čekání předpokládá existenci pomocných prázdných akci, které vkládáme do procesu, abychom dosáhli odsunu zakázané oblasti. Do kterého procesu? Samozřejmě že do toho, který se hlásí o vstup do zakázané oblasti jako druhý. Tento druhý proces(a stejně tak další procesy, pokud jich je více) musí prvnímu procesu poskytnout čas, aby mohl dokončit operace v zakázané oblasti.

11.3.2 Pasivní čekání

Základní myšlenka techniky pasivního čekání je tato: Jsou-li virtuální procesory vytvářeny přepínáním kontextu, které proběhne při přerušení generovaném časovačem, dosáhneme cíle tím, že zamezíme přepínání kontextu. Zákaz lze realizovat ve dvou variantách: jako nepodmíněný nebo jako podmíněný.

11.4 NEPODMÍNĚNÝ ZÁKAZ PŘEPÍNÁNÍ KONTEXTU

Nejsilnější, avšak zároveň nejméně vhodný způsob realizace zákazu přepnutí kontextu, je vydání zákazu přerušení vůbec, protože je tím znemožněno i přerušení od časovače. DI (Disable Interrupt) a DI (Enable Interrupt).
Těchto funkcí však můžeme používat jen při psaní ovladačů periferních zařízení resp. při ovládání přerušovacího systému. Zákaz přerušení DI, tj. , odblokování celého přerušovacího systému, je možno vydat jen na minimální, nezbytně nutnou dobu, abychom neztratili přerušení od nezávisle pracujících periferií. Podrobněji je o tom pojednáno v kapitole o přerušovacím systému.
Druhá možnost spočívá v zákazu přepnutí kontextu, který si může vynutit proces. Ktomuto účelu jsou některé systémy vybaveny dvojicí funkcí Lock a Unlock. Proces, který vydá přikaz Lock způsobí ´uzamčení´ procesoru pro své vlastní potřeby na úkor jiných procesů a instrukcí Unlock dává procesor k dispozici ostatním procesům.
Funkce Lock a Unlock odpovídají přesně dvojici operacíK a Z. Na rozdíl od DI a EI však během uzamčení normálně funguje přerušovací systém, takže může například probíhat přenos dat po sériových linkách do té úrovně, pokud je příslušné zpracování součástí ovladače sériových portů.
I zde však je nutno dodržovat zásadu, že process uzamyká procesor na co nejkratší dobu, jaká je nezbytná pro zpracování určitého úseku kódu bez případné intervence jiného procesu. Pak je třeba procesor uvolnit a okamžitě o něj třeba hned požádat, následuje-li další nedělitelná sekvence instrukcí. Toto uvolnění je nezbytné proto, aby proces příliš nemonopolizoval procesor, což je neslučitelné s prioritní strategií přidělování procesoru jednotlivým procesům. Instrukcí Lock může totiž proces i s tou nejnižší prioritou dosáhnout na libovolně dlouhou dobu toho, aby byl procesor přidělen procesům s vyšší prioritou.
Současně vidíme další nevýhodu nepodmíněného zákazu přerušení: zbytečně se zastaví i ty procesy, které nemají nic společného se zakázanou oblastí do níž vstoupil první proces a tedy by čekat vlastně nemusely. To je důvodem pro zavedení techniky podmíněného zákazu přepínání kontextu.

11.5 PODMÍNĚNÝ ZÁKAZ PŘEPÍNÁNÍ KONTEXTU

Základní myšlenka podmíněného zákazu přepnutí kontextu spočívá v tom, že pro různé zakázané oblasti budou definovány a používány různé synchronizační prostředky. Pak lze postupem analogickým předešlému vynutit si zákaz přepnutí kontextu jen pro omezenou množinu procesů. Proto podmíněný zákaz. Procesy které nemají se zakázanou oblastí, která je právě " ve hře" nic společného, nemusejí být omezovány víc, než je dáno obecnými pravidly přidělování procesoru v daném systému plánování procesů.
Dva nejjednodušší synchronizační nástroje pra realizaci podmíněného zákazu přepnutí kontextu procesů představují Semafor a Signál.

11.6 SEMAFOR

Semafor je v podstatě dvouhodnotová proměnná, jejímž úkolem je udržovat informací o stavu jí přiřazené zakázané oblasti, tuto informaci, zda do ní některý proces vstoupil. Nechť tedy

například hodnota OBSAZENY znamená, že zakázaná oblast je "obsazena", hodnota VOLNY, že je do ní možno vstoupit.
Pro práci se semafory zavedeme dvě systémové operace SECURE a RELEASE jakožto konkrétní implementace obecných synchronizačních operací Z a K. Definujme nyní datový typ Semafor a operace SECURE a RELEASE.

11.7 SIGNÁL

Od signálu budeme očekávat především odstranění nežádoucí režie spojené s aktivním čekáním. Přesunutím kontroly a řízením operací se signály do nadřazené úrovně se vytváří možnost vytvořit prostředek obecnějšího použití a vlastností než má semafor.
Problematikou synchronizace se totiž doposud zabýváme na základě modelové situace představované kritickou, resp. zakázanou oblastí v souvislosti s problémem výlučného přístupu. O něco obecnější situaci, kdy je třeba synchronizovat činnost dvou neb více procesů představuje úloha, kdy jeden z procesů musí čekat než jiný proces provede nějakou akci na jejímž základě může teprve pokračovat v činnosti, Přitom první proces nemusí být vázán na, stav druhého procesu a může pracovat na něm nezávisle. Může tedy tuto akci provést v předstihu, v okamžiku, kdy na ni ještě druhý proces nečeká a ten se musí dovědět o tom, že akce byl, provedena v okamžiku, kdy se dostane do místa, stavu, kdy bude potřebovat její výsledek a kdy na ní bude muset čekat, pokud ne bude provedena.
V této situaci vystupuje jeden a procesů jako zákazník, klient, druhý proces jako obsluha, server (Latinsky servus znamená otrok). Provedením služby vytváří server událost, na níž čeká klient. Záleží na situaci, byla-li tato o událost klientem vyžádána anebo jde-li o událost na niž může čekat více procesů, více klientů.
Událost budeme reprezentovat synchronizačním nástrojem signál. Pro jeho zpracování vytvoříme tři operace. SEND pro ohlášení události, aktivaci signálu, WAIT pro čekání na aktivaci signálu a INIT pro počáteční nastaveni jeho hodnoty.

11.8 KRITICKÉ SEKCE, VZÁJEMNÉ VYLOUČENÍ

Kritická oblast programu je taková oblast programu, která má být provedena při vzájemném vyloučení s jinou kritickou oblastí jiného programu. Realizovat kritické sekce tedy znamená synchronizovat prováděni příkazů tak, aby došlo k požadovanému vzájemnému vyloučení. Petriho síť specifikující tuto úlohu je znázorněna na obrázku OBRxxi. Synchronizace je založena na této myšlence. Každý a procesů Pi odebere při přechodu VSTUPi ae strážního místa SM tečku. Pokud není v místě SM tečka, přechod se nemůže realizovat. Vidíme, že žádné dva z příkazů KSi se nemohou provést současně, takže jimi řízené procesy jsou vzájemně vyloučeny.
V programech je často třeba uplatnit princip kritických sekcí na více místech a my budeme nazývat všechny kritické sekce, které mají být provedeny při vzájemném vyloučení, sdruženými kritickými sekcemi.

11.9 PRODUCENT - KONZUMENT

V této úloze jde o asynchronní komunikaci procesů pomocí společné paměti konečné délky (výraz asynchronní znamená, že oba procesy na sobě při předávání dat nečekají).
Schéma úlohy je uvedeno na na následujícím obrázku. Producent a konzument mají společnou vyrovnávací paměť s kapacitu 3 položky.

```
                1          1(4,7 ..)
Producent ----->           ------> Konzument     3(6,9..)  2        2(5,8..)
```

Proces Producent sekvenčně vytváří jednotlivé položky očíslované 1 , 2, 3, 4, 5, 6, 7, 8, 9, . . . atd a zapisuje je postupné do vyrovnávací paměti do míst v pořadí 1 , 2, 3, 1 , 2, 3, 1 , 2, 3, . . . atd . Proces Konzument v témž pořadí tyto položky odebírá a zpracovává.
Synchronizaci procesů Producent a Konzument je třeba zabránit tomu, aby proces Konzument nepředběhl při výběru proces Producent, tj. aby neodebral ještě nevytvořenou položku. A naopak aby proces Producent nepředběhl proces Konzument "o jedno kolo", tj. aby nepřepsal ještě neodebranou položku.
Specifikace úlohy Producent-Konzument je na OBRxx2. Princip modelu spočívá v použití dvou pomocných míst VOLNÉ (případně ČEKÁ) a OBSAZENÉ, v nichž počet teček odpovídá počtu volných, případně obsazených míst. Proto proces Producent odebere před zápisem jednu tečku z místa volné a po zápisu ji přidá do místa obsazené . Podobně proces Konzument odebírá před čtením z vyrovnávací paměti tečku z místa OBSAZENÉ, případně zde čeká. Tečku vrátí do místa VOLNÉ.
Úloha p-K má výrazné praktické aplikace - je podstatou každé asynchronní komunikace procesů. například v ovladačích sekvenčně pracujících periferii, u kterých má smysl vyrovnávat rozdílnou rychlost vlastního zařízení a toku požadavků na provedení operace.

11.10 ČTENÁŘI - PISAŘI

Často se vyskytuje tato úloha v databázových systémech. Její podstatu lze formulovat takto: n procesů = čtenáři a m procesů - písaři má přistup ke společné datové struktuře. Čtenáři mají tento přístup jen operací čtení, písaři operaci zápisu. Je třeba synchronizovat tak, aby mohly operace čteni probíhat současně, ale přitom byly vzájemně vyloučeny s operacemi zápisu. Současné musejí být vyloučeny i zápisové operace.
Procesy je třeba synchronizovat tak aby operace čtení mohly probíhat současně, ale aby přitom byly vzájemně vyloučeny s operacemi zápisu. Vzájemně musí být také vyloučeny všechny zápisové operace. Jinak řečeno každý písař musí počkat na dokončení všech operaci čtení a zápisu.
Specifikace této úlohy Petriho sítí je na obrázku OBRxx3. Před každou operací čtení odebere čtenář jednu tečku ze strážního místa ZÁPIS_MOŽNÝ a po skončení operace ji sem vrátí. Každý proces PÍSAŘ odebere ze strážního místa před zápisem celkem n teček (n je počet 'čtenářů) .
Teček je tolik, kolik je čtenářů, aby mohli všichni najednou číst. Nespravedlivé je toto řešení a hlediska písaře, když čeká pak čtenáři mohou zahajovat operace čtení. Zároveň toto řešení předpokládá, že písařů není dvakrát více než čtenářů. Pak by nebyly vyloučeny například dvě operace zápisu, nebo by jinak musel proces písař odebírat více teček, než je čtenářů.

11.11 SOUBĚH (RENDEZ-VOUS)

Podstatou souběhu je společné provedení určitých úseků programů dvěma procesy. Procesy je nutno synchronizovat tak, aby před společným úsekem (provedením společných úseků) počkal jeden proces na druhý a ve společném úseku pak postupovaly oba procesy stejnou rychlostí. Některé programovací jazyky mají pro vyjádření souběhu přímo zabudovány prostředky, například ADA.
Specifikace této úlohy pomocí Petriho sítě má navodit představu, že zápis společného úseku je začleněn do programu řídicího jeden z procesů. V postupovém prostoru pak části odpovídající souběhu odpovídá úsečka, jejíž směrnice je vždy i, což z hlediska procesů vypadá jakoby nedocházelo k přepínáni kontextů.
Aplikace tohoto modelu se vyskytuje nejčastěji u paralelních procesů probíhajících v distribuovaném prostředí. Pro toto prostředí je charakteristická spolupráce více procesů pomoci komunikačních kanálů bez využívání společné paměti.

11.12 UVÁZNUTÍ A STÁRNUTÍ

K uváznutí procesu dochází tehdy, jestliže vlivem použití synchronizačních nástrojů nemůže proces dále postupovat. V této situaci se může ocitnou obecně více procesů.
Ke stárnutí procesu dochází tehdy, jestliže vlivem nesprávně použitých synchronizačních nástrojů některý z procesů je více předbíhán ostatními procesy a nepokračuje v postupu.
K uváznuti procesu může typicky dojít při nepozorném,. resp. nesprávném použití synchronizačních operaci, nepřiklad tehdy, když při výstupu z kritické oblasti jeden z procesů neuvolni semafor. Následkem toho druhý proces před vstupem do kritické oblasti uvázne. Typicky vznikají tyto situace při překryti dvojic sdružených kritických oblastí, kde vzniká požadavek ne současné použiti dvou společných prostředků.
Petriho sítě a jejich použíti pro znázornění synchronizace.
Jedním z nejdůležitějších pojmů, který se objevuje nejenom v souvislosti s paralelními výpočetními architekturami ale například v některých diskrétních simulačních jazycích, je pojem proces. jako dekompoziční jednotka systému tvořeného posloupností událostí.
Tato dekompoziční jednotka je zvláště vhodná pro takové systémy, v nichž se objevují synchronní i asynchronní činnosti v rámci vnitřního paralelismu jednotlivých často násobných, komponent systému. V těchto systémech, označovaných jako paralelní systémy, lze identifikaci elementárních procesů, případně stanovením jejich hierarchií a určením 2působu jejich komunikace, podstatně zjednodušit popis i značně komplexních systémů.
Úvahy o isomorfismu mezi subsystémy objektů reálného světa ne jedné straně a jejich popisem a počítačovým modelem na straně druhé jakož i praktické zkušenosti ukazují, že k současným i perspektivním výpočetním systémům je výhodné přistupovat právě jako k paralelním systémům.
Pojem paralelní systém je odvozen od způsobu dekompozice diskrétních systémů na relativné samostatné funkční jednotky procesy, které probíhají paralelné v čase a které spolu komunikují v rámci vzájemných interakcí.