INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

VYSOKÁ ŠKOLA BÁŇSKÁ – TECHNICKÁ UNIVERZITA OSTRAVA
FAKULTA STROJNÍ

PROGRAMOVÁNÍ ŘÍDÍCÍCH SYSTÉMŮ

Procesy, paralelní procesy, souběžné zpracování

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

13 PROCESY, PARALELNÍ PROCESY, SOUBĚŽNÉ ZPRACOVÁNÍ

Obsah kapitoly:

Definujte multiprocessing, distribuované zpracování

Popište souběžné zpracování, paralelní procesy, systémy reálného času

CÍL:

Po prostudování tohoto odstavce budete umět:

- definovat multiprocessing, distribuované zpracování
- popsat souběžné zpracování, paralelní procesy, systémy reálného času

13.1 PROCESY, PARALELNÍ PROCESY, SOUBÉŽNÉ ZPRACOVÁNÍ

V následujících přednáškách budeme hovořit o problematice zpracováni sobžně pracujících úloh na výpočetním systému. Slovo souběžně je použito jako překlad anglického »concurrent« . Concurrent programming, concurrent processing. Co to znamená ?
Concurrent programming , neboli technika programováni souběžně pracujících úloh je technika tvorby programů, které umějí dělat více věcí »v témže čase <
Jak dosáhnout na počítači současného provádění více úloh současně? V principu ¨dvěma způsoby:

a) na jednom procesoru nebo
b) na více procesorech

Programování takových úloh, které vyžadují souběžné pracující programy souvisí dále se dvěma pojmy:

a) programování reálného času (real time programming)
b) Paralelní prgramování

13.2 SYSTÉMY REÁLNÉHO ČASU

Jsou to takové systémy, kde časové požadavky mají podstatnou váhu a hrají důležitou roli při specifikování funkcí systému. Časové požadavky se mohou projevit:

a) Jako požadavky na rychlost reakce na podněty přicházejecí z okolí počítače (typické pro řídicí systémy)
b) Jako podmínky stanovující časový limit pro zpracování nějakých úloh
c) Jako požadavky na automatické časové startování úloh a to buď cyklicky nebo v zadaných absolutních časech

Příklady:

- každý operační systém,i monoprogramový, obsahuje funkce, které mají RT-charakter: například IO subsystém.
- systémy zpracování transakcí
- systémy řízení průmyslových technologií
- vestavné systémy (embeded systems) . Jsou to systémy, které se zabudovávají do nejrůznějšíchh technických výrobků a » prudce vylepšují jejich inteligenci « , jako jsou auta, pračky, CD disky atd.
- komunikačni systémy: telefonní ústředny, satelitní komunikace

13.3 PARALELNI SYSTÉMY

Jsou to systémy implementované na počítačích, resp. výpočetních systémech, se dvěma nebo více současně pracujícími procesory. Ale pozor: podmínkou pro to, abychom mohli považovat aplikaci za paralelní je, aby řešení nějaké logicky ucelené úlohy bylo distribuováno alespoň mezi dva procesory.
Řekneme-li tedy, že aplikace je paralelní, říkáme cosi o implementaci, řekneme-li, že jde o RT-aplikaci, říkáme něco o jejich vlastnostech požadovaných z hlediska časového zpracování. Přesto je zřejmé, že mezi paralelními a RT-systémy existuje souvislost daná minimálně tím, že bychom asi netvořili paralelní systémy, pokud bychom se nedostávali do časových potíží. Avšak i z metodického hlediska, jak pokud jde o analýzu úloh tak jejich implementaci mají oba přístupy mnoho společného.
Společným podstatným rysem obou typů systémů je jejich požadavek na spolehlivost a to jak v oblasti HW, tak v oblasti SW. Spolehlivost HW je dnes vysoká, daná jednak vysokou úrovní konstrukčních technologií, jednak dalšími možnostmi zabudovávání kontrolních obvodů a mechanismů na úrovni HW. V SW oblasti je situace o mnoho složitější. Neustále do

ní totiž vstupuje člověk se svými novými programy, které jsou plné jeho chyb: ať již jde o chyby ve specifikaci, její implementací atp. Na paralelních systémech pak je obtížným problémem ladění, protože je vlastně nemožné udržet kontrolu nad stavem všech paralelně, na různých procesorech nestejným tempem probíhajících úloh.
Návrh souběžné pracujících programů.
Jednoprocesorové, tj. normální počítače, pracují sekvenčně a většina programovacích jazyků dovoluje formulovat výpočetní postupy jako sekvenčně prováděné posloupnosti operací.
Souběžně pracující programy mají části, kterým se říká PROCESY, a které mohou pracovat souběžně. Avšak procesy samy jsou sekvenční. Souběžné provedeni procesů pak může být skutečně paralelní, nebo pouze quasiparalelní.
PROCES je část programu (nebo systému ,pak to může být i program), která je prováděna přísně sekvenčně (instrukce zapsaná v textu programu dále od začátku se provede v čase později) ale která může být provedena souběžně paralelně (reálně nebo virtuálně=zdánlivě) s ostatními procesy programu.
Rozděleni počítačových architektur z hlediska paralelismu.
Monoprocesorové architektury - dovolují pouze quasiparalelismus, paralelně na nich provozované systémy jsou označovány jako systémy pracující ve sdílení času (time sharing) nebo multiprogramové systémy, vlastní procesy pak slovem PROCES, nebo TASK nebo THREAD.
Víceprocesorové architektury

- MULTIPROCESSING provádí souběžně pracující procesy na více procesorech, které mohou přistupovat ke společně sdílené paměti a samy mohou mít vlastní lokální paměť.
- DISTRIBUOVANÉ ZPRACOVÁNÍ (distributed processing) zde jsou procesy prováděny na oddělených procesorech bez možnosti přístupu ke společně sdílené paměti. Paměť, pouze lokální, mají jednotlivé procesory a výměnu dat je nutno provádět pomocí zasílání zpráv po komunikačních kanálech.Typickým představitelem jsou zde systémy založené na transputerech.

Každodenní zkušenost nám dokazuje, že svět, který nás obklopuje je světem paralelních dějů. Dějů, které probíhají v čase současně a vzájemně se ovlivňují. Proto při řešení úloh modelujících děje probíhající v reálném čase hraje podstatnou úlohu paralelní zpracování (paralelismus).
Jako systémy pracující v reálném čase označujeme počítačové aplikace, které lze rozdělit zhruba do dvou skupin:

a) Nepřetržitě pracující výpočetní, informační nebo výrobní počítačové systémy, v nichž ke vstupu, zpracování a výstupu dat dochází v čase náhodně, částečně nepředvídaně a z více různých míst nebo zdrojů (terminály, jiné počítače atp.).
b) Systémy řízení technologických pochodů, které vyhovují předchozí charakteristice a navíc mají daleko přísnější požadavky na rychlost zpracování jednotlivých požadavků

Zatímco v běžných počítačových úlohách a výpočtech vystačíme s jediným programem, výpočetním procesem, v těchto systémech je zapotřebí ke splnění požadovaných funkcí ne jednoho programu, ale více programů, které musí pracovat souběžně. Přitom které programy a kdy budou pracovat souběžně není předem známo a závisí to na tom, jak se situace vyvíjí v "okolí" výpočetního systému, tj. například v technologii, která je systémem řízena.

13.4 VÝPOČETNÍ PROCESY

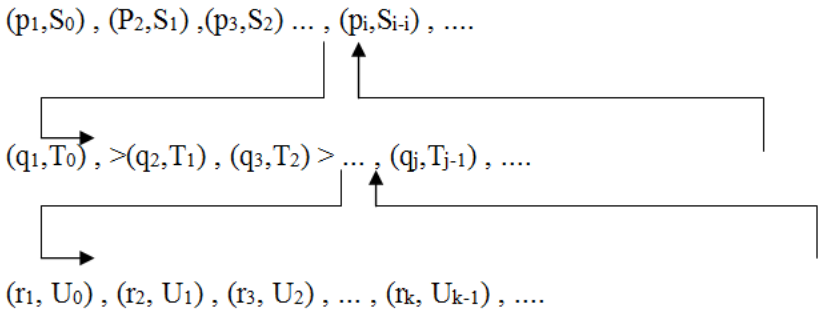
Používáme-li jednoprocesorový počítač, pak se určitá sekvence příkazů programů v nějakém programovacím jazyce (např. Pascalu, atp.) provede sekvenčně, jako posloupnost výpočetních akcí, operaci. Takové posloupnosti výpočetních akcí říkáme sekvenční výpočetní proces, krátce proces.

13.4.1 Rozdíl mezi procesem a programem

Technický objekt, část počítače, která výpočetní akce provádí se nazývá procesor. Každá výpočetní akce je určena příkazem a stavem objektů (proměnných), které v programu vystupují. Výpočetní akci je možno pak formálně popsat uspořádanou dvojici (p, S), kde p je příkaz a S stav objektů před provedením příkazu.

13.4.2 Popis výpočetní akce

(p, S)
p - Příkaz (instrukce programu)
S - Stav datových objektů před provedením
Provádění programu, neboli proces, lze pak chápat jako posloupnost uspořádaných dvojic
(P1, So), (P2, Si), ..., (pi, Si-1), ...
Tato posloupnost může být teoreticky nekonečná. Proces má dynamický charakter. Jednotlivé prvky posloupnosti se sekvenčně generují, stav Si vznikne provedením akce (pi, Si-1) . Provádí-li se některá akce procesu, říkáme, že proces běží. Naproti tomu program, ze kterého vybírá procesor příkazy určující jednotlivé akce se nemění. I když je možno si představit, že pi jsou příkazy vyššího programovacího jazyka, například Pascalu, bude pro pochopení dalšího výkladu lepší představovat si, že pi jsou instrukce na úrovni strojového kódu.
Dnes převážně používané počítače - až na výzkumné případy (např. stroje s objektovou architekturou) a speciální multiprocesorové stroje - jsou jednoprocesorové počítače von Neumannovy koncepce a tedy v principu prostředky sekvenční na úrovni architektury strojového kódu. Říkáme také, že tyto počítače mají vN architekturu.
Je proto na místě otázka, co pak rozumíme paralelismem provedení na sekvenčním počítači vN architektury. Již jsme naznačili, že to bude paralelismus zdánlivý, virtuální, a že ho docílíme rychlými přechody od řešení jedné úlohy (programu) k druhé, jak si hned ukážeme dále.
Nechť je program charakterizován dvojicí < P, S > kde P je konečná množina všech příkazů programu a S množina datových objektů, které se mohou měnit prováděním programu.
Mezi těmito-datovými objekty má význačné postavení určitá podmnožina a tvořená obsahy všech datových registrů procesoru, a která se mění provedením každé strojové instrukce. Tato množina údajů a se nazývá kontext programu a je tedy vlastně S = σ + s.
Souběžné, konkurenční, provádění více programů <P,S> , < Q, T> , <R, U> jakožto para1e1ních procesů, zpracovávaných jedním skutečným procesorem, pak může například vypadat tak, jak je znázorněno na dalším schématu.



13.5 PŘEPNUTI KONTEXTU

Vidíme že posloupnost provedení příkazů reálným procesorem je vlastně následující:
(p1, S0) , (p2, S1) , (q1, T0) , (r1, U0) , (r2, U1) , (r3,U2) , (q2,T2) , (q3,T2) , (p3,S2) , ...
Je přirozené, že při přepínání úloh je nutno uschovat kontext programu a při opětovném přechodu k řešení úlohy zajistit jeho obnovu tak, aby mohl program navázat přesně na stav v

němž bylo jeho prováděni přerušeno. Mechanizmu přechodu od řešení jedné úlohy k druhé se proto také říká přepínání kontextu programu.
Prakticky je přepnutí kontextu umožněno mechanismem hardwarového přerušení. Přerušovací systém počítače umožňuje přerušit prováděni běžícího programu na základě tzv. vnitřního přerušeni odvozeného od hodinových pulsů generovaných hardwarem počítače anebo na základě vnějších přerušeni generovaných souběžně pracujícími periferními zařízeními. Tato periferní zařízeni vydávají přerušovací signál pří dokončeni operací, které byly vyžádány procesorem. .
Tak může na jednom počítači probíhat souběžně několik paralelních procesů, které používají tytéž technické prostředky (periferní zařízení, operační paměť atp.)
A zde vzniká jeden z obecných problémů paralelismu: Korektní použití společného prostředku několika paralelními procesy. To znamená, že při použití společných prostředků více procesy nesmi dojít k tomu, aby vlivem přepínání kontextu mezi jednotlivými procesy došlo k nesprávné činnosti jednoho nebo více z nich. Je zřejmé, že jde o problém času, časové závislosti procesů, který by nevznikl kdyby mohl být proces proveden samostatně.

13.6 PROBLÉM ČASOVÉ ZÁVISLOSTI

Při zpracováni paralelních procesů se mohou vyskytnout dvě základní situace. Buď jsou tyto procesy (alespoň dva) vzájemně úplně nezávislé v tom smyslu, že nijakým způsobem nespolupracují, tj. nepodílejí se o společná data atp. Anebo je tomu naopak: existuje nějaký společný objekt, jako např. periferní zařízení, datová struktura, soubor atp., jehož společným používáním se mohou akce různých procesů vzájemně ovlivňovat.
Intuitivně je zřejmé, že časově závislé chyby jsou důsledkem nevhodného používání společných objektů paralelních procesů. Odstraněni těchto chyb proto musí spočívat v koordinovaném "promyšleném" používání společných objektů, což v konečném důsledku znamená přijmout určitá omezení v přístupu k nim.
Tato omezení je možno vyjádřit např. ve formě Bernsteinových podmínek. Formulujme Bernsteinovy podmínky pro dva procesy řízené příkazy P a Q, jejichž jedinými společnými objekty jsou nějaké proměnné. Bernsteinovy podmínky požaduji, aby procesy byly deterministické, tj. vzájemně na sobě nezávislé.

1 .	R(P)	&	W(Q)	-	O
2.	R(Q)	&	W(P)	=	O
3.	W(P)	&	W(Q)	-	O

Zde je:

R(S) množina proměnných	užitých v příkazech S	v operaci čtení z paměti
W(S) množina proměnných	užitých v příkazech S	v operaci zápisu do paměti

O prázdná množina
Vysloveno explicitně: Bernsteinovy podmínky požadují, aby množina společných proměnných užitých v příkazech čteni nebo zápisu jednoho procesu byla disjunktní s množinou proměnných užitých v příkazech zápisu druhého procesu, neboli aby procesy spolu nekomunikovaly prostřednictvím poměných k nímž má alespoň jeden z nich právo zápisu.
Tyto podmínky jsou tedy velmi silné, omezující a pro nás prakticky nezajímavé. Nás totiž především řešení právě situaci explicitního paralelismu, což je situace běžná při programování řídicích systémů a zde se bez přístupu různých procesů ke sopečných proměnných neobejdeme: právě naopak na něm je většinou založena idea dekompozice systému řízeni.
Bernsteinovy podmínky stačí k tomu, aby procesy byly deterministické. To znamená, že posloupnost akcí kterékoliv z procesů je jednoznačné určená počátečním stavem objektů So a příslušným programem. Takové procesy jsou pak defacto nezávislé, alespoň pokud jde o možnost vzájemného nežádoucího ovlivnění dat, které zpracovávají.

U paralelních procesů, které jsou nedeterministické, naopak dochází k jejich ovlivňováni s postupem času a tato vlastnost se nazývá časová závislost procesů.
Vlivem časové závislosti procesů pak může docházet k chybám, např. tím, že dojde k současnému použití společných proměnných oběma procesy.

13.7 VLASTNOSTI OS REÁLNÉHO ČASU

Nejdříve musíme přesněji specifikovat, co máme na mysli, hovoříme-li o operačním systému reálného času, resp. o vlastnostech nutných pro provozování RT-aplikací.
Budeme říkat, že operační systém má vlastnosti operačního systému reálného času, jestliže splňuje následující tři požadavky:

- deterministické chováni
- rychlá reakce na externí události
- rychlé reakce na vnitřní události

13.7.1 Deterministické chování

Jednou z nejdůležitějších charakteristik OS reálného času je zaručená krátká doba odezvy. O deterministickém chování pak hovoříme tehdy, jestliže doba odezvy je zaručena i při maximální zátěží systému zapříčiněné požadavky aplikačních úloh.
Rychlá odezva na externí události
Aby mohl systém reagovat na vnější událostí v nejkratším možném čase, musí být zajištěno efektivní řízeni procesů pod OS. Toho lze dosáhnout:

- přiřazením vhodného,resp. potřebného, programu vnější události (vnějšímu přerušeni)
- aktivaci tohoto programu v co nejkratším možném čase
- uložením procesu (odpovídajícího programu) trvale v paměti, jako programu rezidentního v paměti.

Toho lze dosáhnout pouze při použití prioritního systému plánování procesů využívající strategii tzv. preemptivního plánování.

13.7.2 Rychlá odezva na vnitřní události

Rychlá aktivace procesu není dostatečnou podmínkou pro rychlé zpracování RT úlohy. K tomu je třeba:

- účinné komunikační a synchronizační nástroje (funkce) pomocí nichž mohou být různé procesy rychle synchronizovány
- účinné prostředky pro rychlou výměnu dat mezi procesy

Standardní OS UNIX System V tyto požadavky nesplňuje anebo alespoň ne v takové míře, aby mohl být použit bez omezeni pro nejrůznější RT-aplikace.
Operační systém SORIX, který vyvinula fa Siemens AG obsahuje rozšířeni pro oblast reálného času nad rámec standardu UNIX umožňující dosaženi deterministického chování a tím pro aplikaci řízení v reálném čase.
Přehled RT-funkcí OS SORIX
RT - vlastnosti OS SORIX jsou založeny především na přerušitelných rutinách (funkcí) jádra (operačního systému) a na účinném řízení procesů, v dalším sledu pak na silných funkcích pro komunikaci mezi procesy, časovacích funkcích a na systému pro obsluhu souborů, který je optimalizován s ohledem na použiti v RT - prostředí.

- Přerušitelné rutiny jádra
  - proces může být přerušen i během volání resp. využívání systémových funkcí
- Procesy rezidentní v paměti
  - umožňuje se, aby časově kritické procesy byly trvale uloženy v operační paměti i v době,kdy nejsou aktivní. To umožní jejich rychlou aktivaci.

- Řízení procesů technikou preemtivního plánování
  - umožňuje aplikovat techniku aktivního procesu čistě na základě jeho priority s betzprostředním přerušením procesu s nižší prioritou aniž tento vyčerpal tzv. časovou dávku (time-slice).
- Mechanismus událostí s vícenásobným čekáním
  - umožňuje účinnou komunikaci mezi procesy s možností čekání na libovolnou kombinaci asynchronních událostí (multiple waiting).
- Rychlý semafor (quick semaphore)
  - pro rychlou syncronizaci nezávislých procesů.
- Časové funkce
  - umožňují plánování procesů (akci) v piředem zvolených absolutních časových okamžicích nebo cyklicky zvoleným časovým intervalem
- Optimalizovaný systém obsluhy souborů
  - s funkcemi minimalizujícími přístupové časy vstupně/výstupních operaci a s možností přímého vstupu/výstupu by passing rychlé vyrovnávací paměti (cache bugger). Kromě toho obsahuje OS SORIX další funkce, které sice nemají bezprostřední vztah k RT-funkcím,ale jsou užitečné pro automatizaci úloh.

14 POUŽITÁ LITERATURA, KTEROU LZE ČERPAT K DALŠÍMU STUDIU

[1] Špička, Ivo, Programování řídicích systémů. Studijní opora k předmětu. 2013.
[2] Pavel Herout, Učebnice jazyka C. 1. díl , 5. vyd. České Budějovice : Kopp, 2008 - 271, viii s. ISBN 978-80-7232-351-7 (brož.)
[3] Herout, Pavel, Učebnice jazyka C. 2. díl, 3. vyd. České Budějovice : Kopp, 2007 - vii, s. 272-437 ISBN 978-80-7232-329-6 (brož.)
[4] Qing Li, Caroline Yao, Real-Time Concepts for Embedded Systems, CRC Press; 2003, ISBN 1578201241
[5] Doporučená literatura:
[6] WIRTH, N.: Algoritmy a štruktúry údajov, Alfa Bratislava, 1988.
[7] JOSEPH, M.. Real-Time Systems Specification, Verification and Analysis. London : Prentice Hall, 1996. ISBN 0-13-455297-0.