

Vlastní implementace vláken v reálném režimu x86

Obsah:

1. Cíl
2. Rozhraní jednotky pro podporu procesů a binárních semaforů
3. Práce producenta a konzumenta
4. Implementace podpory více procesů
5. Implementace binárních semaforů
6. Implementace podpory semaforů podle Dijkstra
7. Vestavné systémy a DPMI

Cíl

S použitím modulu podpory "procesů" a vlastní implementace semaforů budeme implementovat řešení problému producent-konzument s těmito vlastnostmi:

- producent bude produkovat libovolné znaky, před vyprodukováním každého znaku bude čekat náhodnou dobu (použijeme generátor pseudonáhodných čísel Turbo Pascalu)
- konzumace každého znaku bude trvat konstantní dobu
- vyrovnávací paměť bude použita jako kruhový buffer, aktuální stav vyrovnávací paměti (její obsazené a volné položky) průběžně znázorníte.

1. Rozhraní jednotky pro podporu procesů a binárních semaforů

Rozhraní poskytuje jednotka *Tasks* pomocí následujících funkcí:

Function InitTasksEnv(TblSz:word):integer;

Provede inicializaci víceúlohového prostředí. *TblSz* udává maximální možný počet nainstalovatelných procesů. Tato funkce se musí volat jako první.

Vrací: 0 - úspěch
-1 - málo paměti
-2 - není i386+ kompatibilní, anebo nemá kompatibilní koprocessor

Function DoneTasksEnv:integer;

Uvolní paměť alokovanou pomocí funkce *InitTasksEnvironment*. Tuto funkci je možné volat pouze po skončení funkce *RunTasks*.

Function InstallTask(address:pointer;StackSz:word):integer;

Nainstaluje proces do paměti, ale nespustí ho. Jednotlivé procesy lze instalovat pouze mezi funkcemi *InitTasksEnv* a *RunTasks*.

Vrací: kladné číslo jako handle procesu, pokud je vše v pořádku
-3 - v tabulce procesů je už plno

Function KillTasks:integer;

Ukončí provádění jednotlivých procesů a ukončí provádění funkce *RunTasks*. Tato funkce provádí danou činnost pouze tehdy, je-li volána v době běhu *RunTasks*. Vždy vrací nulu.

Function RunTasks:integer;

Spustí všechny nainstalované procesy. Nesmí být spuštěno dříve než před voláním *InstallTask*.

Function StartExclusiveAccess(var Semaphor:longint):integer;

Zažádá o výhradní přístup ke společným datům určených proměnnou *Semaphor*. Pokud má tento přístup vyhrazen jiný proces, volající bude zablokován do doby, než mu bude přidělen výhradní přístup pro daný semafor. Každá aplikace může mít aktivní pouze jeden semafor. Lze volat pouze během funkce *RunTasks*. Semafor musí být inicializován pomocí *InitSemaphor*, jinak se daný proces zablokuje a už nebude spuštěn.

Vrací: -4 - pokud semafor neexistuje, v Pascalu nenastane
-5 - pokud už proces má exkluzivní přístup na tomto semaforu
-6 - pokud už je aktivní jiný semafor

Proces smí pokračovat pouze v případech 0 a -5.

Function EndExclusiveAccess(var Semaphor:longint):integer;

Ukončí výhradní používání dat určených proměnnou *Semaphor*.

Vrací: -7 není co končit; buďto je špatný semafor, nebo na něm proces není aktivní

Function InitSemaphor(var Semaphor:longint):integer;

Inicializuje semafor před prvním použitím. Použití neinicializovaného semaforu by ve většině případů vedlo k zablokování procesu. V ostatních by se zablokovala možnost použití funkce *StartExclusiveAccess* u některého z procesů.

Procedure FreezeTasking;

Zastaví přepínání procesů. Je určeno pouze pro nezbytné použití místo instrukce CLI.

Procedure UnFreezeTasking;

Obnoví stav přepínání procesů. Je určeno pouze pro nezbytné použití místo instrukce STI. Pokud je procedura použita po *KillTasks* do prvního přepnutí, akce požadovaná *KillTasks* se neprovede.

Function GetSelfHandle:word;

Vrátí identifikaci procesu. To samé co vrací *InstallTask*.

Function BlockProces(proc:word):integer;

Zablokuje daný proces. Pokud je daný proces aktuální, je provedeno přepnutí na další.

Vrací: -8 – proces neexistuje

Function UnblockProces(proc:word):integer;

Odblokuje zadaný proces.

Function SwitchToProces(proc:word):integer;

Pokusí se přepnout na daný proces. Pokud je zablokovaný, pak se přepne na nejbližší volný.

Poznámka:

Všechny uvedené návratové kódy jsou platné pro všechny funkce, není-li uvedeno jinak. Všechny výše uvedené semaforey jsou semaforey binární.

2. Práce producenta a konzumenta

Procesy producent i konzument používají sdílený kruhový buffer implementovaný jako pole a dva semaforey (implementovány podle Dijkstra) pro plný počet položek a pro volný počet položek. Producent generuje jednotlivé hodnoty po náhodně volené době a konzument je „vybírá“ z bufferu.

Každý z nich vypisuje na obrazovku údaje, se kterými pracoval. Jako obsluha klávesnice je nainstalován třetí proces s nejvyšší prioritou, který vůbec nepoužívá semaforey, čímž je zajištěna ovladatelnost celého systému. Program se ukončí klávesou *Escape*.

Ke správnému zajištění výstupu je použita jednotka *Routines* poskytující procedury k nainstalování české diakritiky v kódu Latin 2, speciálních znaků a nastavení VGA.

3. Implementace podpory více procesů

Informace o jednotlivých procesech jsou udržovány pomocí interní proměnné *TasksTbl*, která je pointer na pole pointerů na kontexty jednotlivých procesů. Při inicializaci pomocí *InitTasksEnv* je nastavena velikost tohoto pole a jeho položky jsou nastaveny na hodnotu *nil*. Velikost pole je počet zadaný parametrem + 1. Poslední položka slouží k uchování hodnot potřebných pro bezchybné ukončení *RunTasks*.

Instalace procesu spočívá v alokování požadované paměti jako zásobníku, získání hodnot důležitých registrů (BP, DS), stavu FPU a zrušení aktivity používání semaforu. Registry SS a SP jsou nastaveny podle alokované paměti, registr Flags je uložen s nastavenou vlajkou povolující maskovatelná přerušení. Vstupní bod procesu je uložen v páru CS:IP. Paměťový prostor, do kterého se nastavují tyto hodnoty, se vytváří dynamicky pomocí *GetMem* a do pole odkazovaného pomocí *TasksTbl* se zapíše adresa této struktury na nejbližší volnou pozici. V popisovači procesu je možné uložit stav FPU platný pro instrukci *FRSTOR*, adresa jednoho semaforu, zásobník, velikost zásobníku a následující registry: AX, BX, CX, DX, ES, DI, DS, SI, CS, IP, SS, SP, Flags.

Inicializace přepínání procesů je realizována funkcí *RunTasks*, která nejprve získá adresu staré obsluhy přerušení 8, vyplní záhlaví procedury *TaskMgr* a nastaví řídicí slovo *TaskMgrStatus* na 2 (zmrazení přepínání). Poté alokuje paměť pro uložení vlastních informací, kterou uloží jako další popisovač procesu na poslední možnou pozici v *TasksTbl*. Dále pomocí služby DOSu nastaví novou obsluhu přerušení 8, zakáže maskovatelná přerušení, odblokuje přepínání procesů a zavolá proceduru *TaskMgr* jako přerušení.

Přepínání procesů zajišťuje výše zmiňovaná procedura *TaskMgr* instalovaná jako nová obsluha hardwarově generovaného přerušení 8. Vždy při jejím vyvolání je nejprve zavolána stará obsluha přerušení, která by měla sdělit řadiči, že je konec přerušení. Dalším krokem je otestování pomocí interní vlajky, zda nebyla vyrušena uprostřed vlastní práce. Pokud ano, tak se obsluha ukončí instrukcí *IRET*. Jestliže tento případ nenastal, je další testovanou hodnotou stavové slovo *TaskMgrStatus*. Podle něj mohou nastat tři různé situace.

<i>TaskMgrStatus</i> = 0	má nastat pokus o přepnutí na další proces
<i>TaskMgrStatus</i> = 1	má se neprodleně skončit přepínání procesů
<i>TaskMgrStatus</i> = 2	neprodleně ukončit obsluhu bez přepnutí procesu

Následuje test na počet dosáhnutí aktuálního kódu od posledního přepnutí procesů. Testuje se podle hodnoty určené konstantou *TicsPerTask*. Teprve po splnění všech uvedených podmínek je možné pokračovat v pokusu o přepnutí na další proces.

Přepnutí na další proces předchází uložení aktuálního stavu registrů přerušeného procesu do příslušného popisovače procesu. Po uložení jeho stavu, tj. všech registrů, které lze uložit, se vyhledá další proces, který buď nepoužívá semafor, nebo nějaký semafor používá, ale je v něm právě aktivní. Díky tomu, že každý proces může mít aktivní pouze jeden binární semafor, nelze se při alespoň jednom nainstalovaném procesu dostat do situace, kdy není možné najít volný proces. Vzhledem k možnostem semaforů však může nastat případ, kdy se proces nepřepne. Procesy se testují podle pořadí, v jakém se instalovaly. Pro dokončení přepnutí se nastaví registry podle hodnot popisovače naplánovaného procesu.

Ukončení přepínání procesů se provádí, jakmile je stavové slovo *TaskMgrStatus* rovno hodnotě 2. V tomto případě se namísto přepnutí procesu obnoví registry z popisovače na poslední možné pozici, kde jsou uloženy hodnoty registrů pro návrat do *RunTasks*. Obnoví se však pouze ty nejnnutnější. Před tím je ale nastavena původní obsluha přerušení jako výchozí. Tj. odinstalování *TaskMgr* z přerušení 8. Po skončení obsluhy instrukcí *IRET* se řízení programu vrátí do funkce *RunTasks*, kde se povolí maskovatelná přerušení. Poté *RunTasks* vrátí nulu.

Na začátku rutiny *TaskMgr* je blízký skok o několik bytů dále. Přeskočené místo je použito pro uložení následujících údajů: adresa stavového slova, adresa původní obsluhy, počet volání, vlajka vlastní aktivity, adresa popisovače prvního procesu, číslo aktivního procesu a počet aktivních procesů. Před začátkem vlastního kódu následuje osmibytový odkládací prostor, takže volající nemusí disponovat vlastním zásobníkem.

4. Implementace podpory binárních semaforů

Semaforem se rozumí 4 byty v následujícím formátu. První dva byty indikují, kdo právě využívá výhradní přístup zaručovaný daným semaforem. Tyto dva byty jsou používány rutinou *TaskMgr*, která sem ukládá číslo procesu. Pokud jsou rovny 0xFFFF, pak je daný semafor volný a může být obsazen. Další dva byty udávají kolik procesů je právě vázáno na daném semaforu. Avšak vzhledem k principu implementace semaforů mají spíše informativní

význam a nejsou pro správnou činnost důležité. Jsou zde spíše pro ladící účely a pozdější rozšíření.

Inicializace semaforu se provádí funkcí *InitializeSemaphore*, která nastaví 4 byty na hodnotu 0xFFFF0000. V popisovači procesu se neukládá hodnota semaforu, ale jeho adresa v paměti.

Funkce *StartExclusiveAccess* nejprve použije stavového slova k pozastavení přepínání procesů. Toto je provedeno pomocí instrukce *XCHG*, takže se nemohou objevit další vedlejší efekty. Poté se otestuje, zda nemáme aktivní jiný semafor, či zda již se zadaným semaforem někdo pracuje. V případě, že semafor není volný, funkce nastaví proměnnou *TaskMgr* udávající počet vyvolání od posledního přepnutí na hodnotu *TicsPerTask*, čímž si zajistí, že při vyvolání *TaskMgr* bude proces přepnut. Jinak se nastaví na semaforu číslo procesu a inkrementují další dva byty. V popisovači procesu se nastaví adresa semaforu, obnoví se *TaskMgrStatus*, tj. odblokuje se přepínání a provede se *INT 8*. Tím je zajištěno, že se proces nedostane ke sdíleným datům příliš brzo.

Funkce *EndExclusiveAccess* pracuje stejně jako *StartExclusiveAccess*, pouze s tím rozdílem, že v době zmrazení přepínání procesů nastaví první dva byty semaforu na 0xFFFF, dekrementuje zbylé dva a v popisovači procesu nastaví adresu semaforu na *nil*. Před tím si však otestuje, zda byl zadán správný semafor.

Tyto semaforey jsou zajišťované jádrem. S jejich pomocí lze naprogramovat libovolnou implementaci semaforů, která bude místo instrukcí *STI* a *CLI* používat *StartExclusiveAccess* a *EndExclusiveAccess*. I když může stále použít *FreezeTasking* a *UnFreezeTasking*.

5. Implementace podpory semaforů podle Dijksra

Rozhraní

Je zajišťováno jednotkou *PVSEMS.PAS* a využívá rozhraní jednotky *TASKS.PAS*.

```
type
  PPVSemaphore = ^TPVSemaphore;
  TPVSemaphore = record
    s: longint;           {pocet blokovanych procesu}
    pos: longint;        {pozice v poli}
    memallocated: word;  {velikost pole}
    proceses: pointer;   {ukazatel na pole blokovanych procesu}
  end;

function CreatePVSemaphore(InitValue: longint; mem: word): PPVSemaphore;
- vytvoří nový semafor s počáteční hodnotou InitValue
- mem je počet bytů určených k uchování informací o zablokovaných
  procesech, jeden proces potřebuje dva byty
- vrací nil když semafor nelze vytvořit

procedure DestroyPVSemaphore(var sem: PPVSemaphore);
- uvolní paměť alokovanou při vytvoření semaforu

procedure P(s: PPVSemaphore);
- operace P

procedure V(s: PPVSemaphore);
- operace V
```

Implementace operace P

Nejdříve je vypnuto přepínání procesů. Poté se sníží hodnota počítadla; pokud je menší než nula, proces se zablokuje a přepne se na další volný proces. Na závěr se povolí přepínání procesů.

Implementace operace V

Jako první je zablokováno přepínání procesů. Poté se inkrementuje hodnota počítadla a je-li nějaký proces blokován, pak se vybere poslední z blokováných procesů a odblokuje se. Na závěr se obnoví přepínání procesů.

6. Vestavné systémy a DPMI

Program je určen pro překladač Turbo/Borland Pascal 7.01 s prostředím MS-DOS. Lze ho spustit v emulátoru DOS Box. Program je zajímavý i z toho hlediska, že i dnes má dědictví procesoru Intel 80186 své místo ve vestavěných systémech – např. viz procesory Turbo86, Turbo186 od VAutomation či HTL80186 od HT-LAB.

V případě naprogramování víceúlohového jádra s podporou semaforů v chráněném režimu pod DPMI serverem by však bylo třeba volat přerušení 31h, které volalo původní rutinu BIOSu. Ale v rámci 16 bitové dosovské aplikace nelze použít výhody TSS. Také by nebylo možné využít možnosti zápisu do kódového segmentu.