Vložené řídicí systémy

Operační systémy reálného času

Pavel Balda
ZČU v Plzni, FAV, KKY

Osnova přednášky

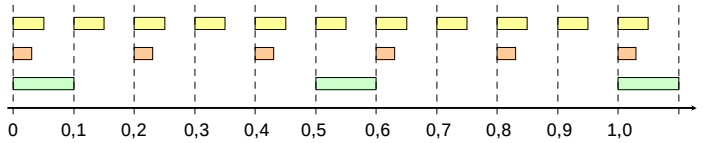
- Plánování (rozvrhování) v OS reálného času
- Windows CE
- Phar Lap ETS
- VxWorks

Co je OS reálného času?

- OS reálného času (RTOS) jsou takové OS, kde čas hraje podstatnou roli
  - Na podnět generovaný externími zařízeními musí reagovat do určitého pevného času
  - Správná reakce, ale pozdě je totéž co žádná reakce!
- Systémy reálného času lze rozdělit na:
  - Systémy pevného reálného času (hard real time) – mezní termíny (deadlines) musí být dodrženy
  - Systémy měkkého reálného času (soft real time) – občasné nesplnění mezního termínu je nežádoucí, avšak tolerované
- Chování systému reálného času musí být predikovatelné a známé předem!
- V systémech reálného času se mohou vyskytovat dva typy událostí:
  - Periodické – objevují se v pravidelných intervalech
  - Aperiodické – výskyt nelze předvídat

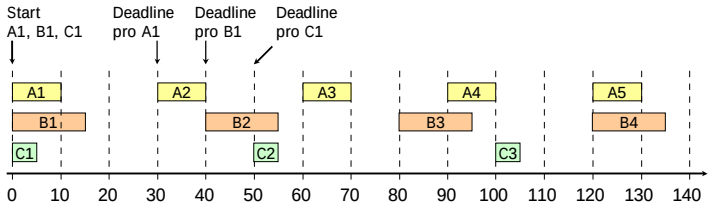
Plánování (rozvrhování) periodických úloh

- Systém může obsluhovat několik posloupností periodických událostí. Předpokládejme:
  - m – počet periodických událostí
  - P_i – perioda i-té události
  - C_i – doba trvání i-té události
- Pak CPU může obsloužit každou událost když platí $U = \sum_{i=1}^{m} \frac{C_i}{P_i} \leq 1$ (1)
  - Systém reálného času, splňující tuto podmínku se nazývá rozvrhovatelný (schedulable)
- Příklad:
  - 3 periodické události s periodami 100, 200 a 500 ms, vyžadující časy CPU 50, 30 a 100 ms
  - Dosazením do (1) dostáváme: $U = 0{,}5 + 0{,}15 + 0{,}2 = 0{,}85 < 1$



Mezní termíny (deadlines) v RT systémech

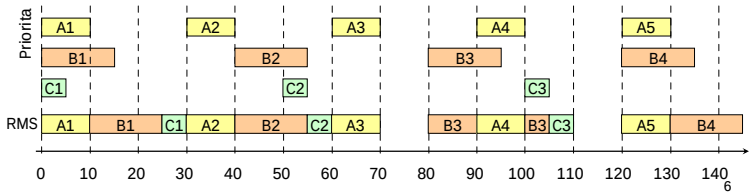
- Deadline (mezní termín) – čas do kterého musí být daná úloha vykonána
- Real-time scheduling (rozvrhování reálného času) – rozvrhování několika úloh „soupeřících“ (competing) o CPU, z nichž některé mají mezní termíny, kterým musí vyhovět
- V případě periodických úloh se často mezním termínem rozumí okamžik, kdy má být spuštěna následující perioda dané úlohy, viz příklad na obr.
  - Systém je rozvrhovatelný, neboť $U = 10/30 + 15/40 + 5/50 = 0{,}808 < 1$



5

Rate Monotonic Scheduling (RMS)

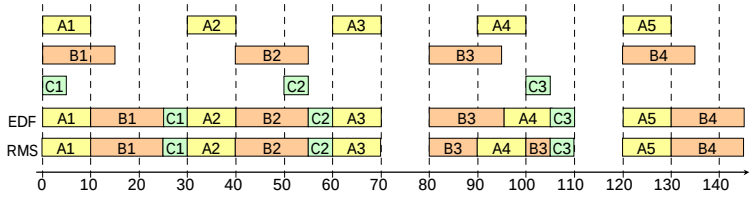
- Statický rozvrhovací algoritmus: Preemptivní spouštění periodických úloh podle priorit
  - Čím je daná úloha častěji spouštěna, tím má větší prioritu (proporcionálně)
- Liu a Layland (1973) stanovili 5 podmínek, za kterých lze algoritmus použít
  - Každý periodický proces musí skončit během své periody
  - Procesy jsou vzájemně nezávislé
  - Každý proces trvá stejnou dobu při každém spuštění
  - Žádné neperiodické procesy nemají deadline
  - Přepnutí procesů (preemption) se provede okamžitě s nulovou režií



6

Earliest Deadline First (EDF)

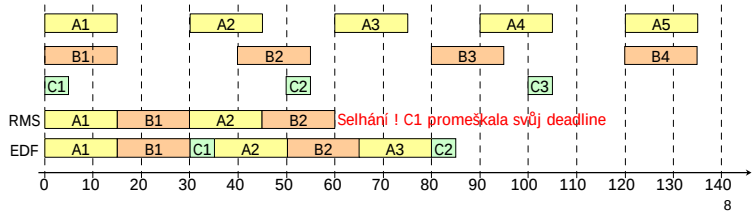
- Dynamický rozvrhovací algoritmus, nevyžadující periodické úlohy ani stejnou dobu trvání úloh při každém spuštění (jako RMS)
  - Každá úloha, které má být přidělena CPU, oznamuje svůj deadline
  - Scheduler vždy spouští úlohu s nejbližším časem deadline
  - K tomuto účelu má scheduler úlohy uložené v seznamu setříděném podle mezních časů
  - Kdykoliv se nějaká úloha dostane do připraveného stavu, scheduler zjistí, zda její deadline není dřív než u právě prováděného procesu. Pokud ano, odstaví prováděný proces a spustí tuto úlohu



7

Rozdíl mezi RMS a EDF

- Podstatný rozdíl se ukáže, vzroste-li zatížení CPU
  - Příklad: úloha A s periodou 30 ms a délkou 15 ms, úloha B s periodou 40 ms a délkou 15 ms, úloha C s periodou 50 ms a délkou 5 ms
  - V tomto případě je $U = 15/30 + 15/40 + 5/50 = 0{,}975 < 1$
  - RMS selže!
- Liu a Layland dokázali, že RMS zaručeně funguje, pokud $U = \sum_{i=1}^{m} \frac{C_i}{P_i} \leq m(2^{1/m} - 1)$
  - V daném příkladě je $U = 0{,}780$
  - Pro m jdoucí do nekonečna je limita rovna $\ln 2$



8

Windows CE

- Windows CE je malý, konfigurovatelný, dobře „propojený“ operační systém reálného času (RTOS).
- Hlavními rysy Windows CE jsou:
  - Multithreadový programový modelu
  - Důmyslná správa paměti
  - Rozsáhlá množina ovladačů
  - Podpora aplikací v reálném čase
  - .NET Compact Framework (od verze 4)
- Není to paradox? Jak může být OS malý a přitom tak bohatý? Díky tzv. Platform Builderu
  - Umožňuje volit komponenty OS nutné pro daného systémového integrátora
  - Nepotřebné ovladače (několik stovek) není třeba začleňovat
- Postup generování image Windows CE
  - Zkonfigurují se potřebné součásti OS, a doplní se o specifické ovladače a aplikace daného systémového integrátora
  - Přeloží se obraz OS (image). Lze jej rovnou poslat do cílového HW
  - Vygeneruje se platformově závislý SDK (software development kit)

Synchronizační objekty ve WinCE (1/3)

- Od verze 3 stejné jako v normálních Windows
- Kritická sekce (`CRITICAL_SECTION`) – Synchronizuje přístup ke kritické sekcí kódu v daném procesu, není však objektem jádra
  - `InitializeCriticalSection()` – inicializuje kritickou sekci
  - `EnterCriticalSection()` – vstupuje do kritické sekce. Pokud v ní je jiný thread, blokuje aktuální thread dokud thread v kritické sekci z ní nevystoupí
  - `LeaveCriticalSection()` – vystupuje z kritické sekce
  - `DeleteCriticalSection()` – ruší kritickou sekci
  - Pomocí kritické sekce se synchronizuje přístup k objektům v příkazu `lock` v C#
- Ostatní synchronizační objekty jsou objekty jádra
  - Mají dva stavy – signalizovaný (signaled) a nesignalizovaný (nonsignaled). Objekt je v signalizovaném stavu, pokud jej nevlastní žádný thread
  - `WaitForSingleObject()` – funkce, která se ukončí, je-li daný objekt v signalizovaném stavu nebo po uplynutí timeoutu
  - `WaitForMultipleObjects()` – funkce, která čeká na první objekt v signalizovaném stavu. (V normálních Windows, umožňuje ještě čekat na všechny objekty v signalizovaném stavu)
  - V systému jsou identifikovány pomocí jednoznačného jména
  - `CloseHandle()` – ukončuje práci s těmito objekty a zavírá jejich handle

Synchronizační objekty ve WinCE (2/3)

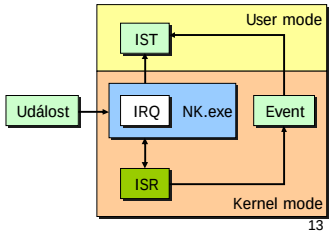
- Mutex – je podobný kritické sekci, ale umožňuje synchronizovat tready v různých procesech.
  - `CreateMutex()` – vytvoří mutex s daným jménem
  - `WaitForSingleObject()` – čeká na vstup do kritické oblasti kódu
  - `ReleaseMutex()` – uvolňuje vlastnictví daného mutexu
  - `CloseHandle()` – ukončuje práci s mutexem zavřením jeho handlu
- Semafor – synchronizační objekt umožňující současný přístup k daném zdroji z `n` míst. Stav semaforu je signalizovaný, když jeho čítač je větší než `0`. Hodnota čítače není nikdy menší než `0` ani větší než `n`.
  - `CreateSemaphore()` – vytvoří semafor s daným jménem, počátečním počtem a maximálním počtem `n`. Windows CE nepodporuje funkci `OpenSemaphore()` jako normální Windows, místo ní se používá `CreateSemaphore()`
  - `ReleaseSemaphore()` – inkrementuje vnitřní čítač semaforu o `1`
  - Spuštění threadu čekajícího na semafor ve `WaitForSingleObject()` dekrementuje čítač semaforu o `1`

Synchronizační objekty ve WinCE (3/3)

- Událost (Event) je synchronizační objekt, jehož stav může být explicitně nastaven na signalizovaný pomocí funkce `SetEvent()`. Existují dva druhy událostí:
  - Ručně resetované (manual reset) – událost, jejíž stav zůstává signalizovaný, dokud není explicitně „shozen“ do nesignalizovaného voláním funkce `ResetEvent()`
  - Automaticky resetované (auto-reset) – událost, jejíž signalizovaný stav se po uvolnění jednoho čekajícího threadu stává automaticky nesignalizovaný. Nečeká-li na událost žádný thread, zůstává její stav signalizovaný
- Dalšími funkcemi událostí jsou:
  - `CreateEvent()` – vytváří událost s daným jménem, příznakem ručního resetování a počátečním stavem
  - `OpenEvent()` – otvírá pojmenovanou událost (jen v normálních Windows a ve WinCE od verze 4.0)

Obsluha přerušení ve Windows CE

- Ovladače ve WinCE se skládají ze dvou částí:
  - Obslužná procedura interruptu ISR (Interrupt Service Routine) – maličká část kódu v jádře instalovaná funkcí **`HookInterrupt()`**
  - Obslužným threadem interruptu IST (Interrupt Service Thread) – další zpracování (významné z hlediska doby trvání). Funkce **`InterruptInitialize()`** mapuje číslo interruptu na událost vytvořenou funkcí **`CreateEvent()`**. IST je normálním threadem vytvořeným funkcí **`CreateThread()`**, který se spouští danou událostí
- Při výskytu interruptu ukládá jádro stav právě prováděného kódu v uživatelském módu, pak volá ISR a podle jejího návratového kódu může jádro voláním **`SetEvent()`** odblokovat daný IST.
- WinCE podporují vnořování interruptů (interrupt nesting)
  - ISR s nízkou prioritou může být odstavena (preempted) ISR s vyšší prioritou
  - Po dokončení obsluhy ISR s vyšší prioritou se pokračuje v obluze ISR s nižší prioritou
  - ISR s vyšší prioritou mají lepší opakovatelnost doby exekuce



Procesy

- Windows CE jsou stejně jako normální Windows víceprocesový (multiprocess) a vícethreadový (multithreaded) OS
- Na rozdíl od stadardních Windows nepodporují více procesorů
- Windows CE omezuje počet současně spuštěných procesů na 32. Do tohoto počtu jsou zahrnuty i standardní procesy WinCE:
  - **`nk.exe`** – jádro OS
  - **`device.exe`** – proces pro spouštění ovladačů zařízení
  - **`filesys.exe`** – proces zavádějící souborový systém
  - **`gwes.exe`** – proces pro grafické uživatelské prostředí (GUI engine)
- Procesy mají adresovatelný prostor 32 MB, kdežto ve standardních windows 2048 MB (2 GB)
- Přepínání kontextů procesů je (záměrně) velmi rychlé.
  - Všechny procesy sdílejí jednu tabulku paměťových stránek
  - Ve Windows 2000/XP má každý proces svou tabulku stránek!

Thready

- Thready – jednotky, kterým je plánovačem přidělován čas procesoru jak ve WinCE, tak i ve standardních Windows
  - Při spuštění je procesu přidělen privátní adresní prostor a jeden thread
  - Thread má svůj vlastní zásobník
  - Další thready se vytváří funkcí **`CreateThread()`**
- Přidělování priorit threadům
  - Mnohem jednodušší než ve standardních Windows, kde patří každý proces do určité prioritní třídy nastavované funkcí **`SetPriorityClass()`**. Jednotlivé thready daného procesu mají prioritu nastavovanou funkcí **`SetThreadPriority()`** jen „jemně“ posunutou od priority třídy procesu.
  - Ve WinCE (od verze 3) je 256 priorit nastavovaných funkcí **`CeSetThreadPriority()`**, nejvyšší priorita je **`0`**, nejnižší **`255`**

Priority threadů (1/2)

- Interval priorit je rozdělen do 4 oblastí
  - **`0`** ... **`96`** – real-time úlohy s vysokou prioritou
  - **`97`** ... **`152`** – interval používaný implicitními ovladači WinCE
  - **`153`** ... **`247`** – úlohy s vyšší prioritou než normální úlohy stanovenou tak, aby neinterferovaly s ovladači zařízení
  - **`248`** ... **`255`** – běžné uživatelské aplikace (non-real-time)
- Implicitní priority jednotlivých ovladačů WinCE jsou uvedeny v následující tabulce
  - OEM (Original Equipment Manufacturer) může priority změnit při generování image WinCE
  - Dále mohou být některé priority změněny nastavením příslušného klíče v registry
  - Všechny klíče začínají **`HKEY_LOCAL_MACHINE`**, který je v tabulce vypuštěn

Ovladač	Priorita	Klíč v registry
USB Function Controller	**`100`**	-
USB OHC Driver	**`101`**	**`Drivers\BuiltIn\OHCI\Priority256`**
USB UHC Driver	**`101`**	**`Drivers\BuiltIn\OHCI\Priority256`**
Serial Port Driver	**`103`**	**`Drivers\BuiltIn\Serial\Priority256`**

Priority threadů (2/2)

- ... Pokračování

Ovladač	Priorita	Klíč v registry
PC Card Socked Driver	105	Drivers\PCMCIA\Priority256
Touch Screen Driver	109	Drivers\BuiltIn\Touch\Priority256
IRSIR Driver	110	Comm\Irsir1\Parms\Priority256
NDIS Driver	116	Drivers\BuiltIn\NDIS\Priority256
EDBG Driver	130	-
CxPort Driver	132	Comm\Cxport\Priority256
Keyboard Driver	145	Drivers\BuiltIn\Keybd\Priority256
IR Comm Driver	148	Drivers\BuiltIn\IrComm\Priority256
TAPI/Unimodem Driver	150	Drivers\Unimodem\Priority256
WaveDev Driver	249	Drivers\BuiltIn\WaveDev\Priority256
WaveAPI Driver	250	Drivers\BuiltIn\WAPIMAN\Priority256

17

Řešení inverze priorit

- Inverze priorit nastává v případě, že thread TL s nízkou prioritou vlastnící kritickou sekci, mutex nebo semafor blokuje thread TH s vysokou prioritou. Mezitím může běžet několik threadů TM se střední prioritou
  - V takovém případě není zaručena doba odezvy threadu s vysokou prioritou!
- Windows CE řeší inverzi priorit do hloubky 1
  - Po zablokování threadu TH s vysokou prioritou threadem TL s nízkou prioritou, zvýší WinCE prioritu TL na prioritu TH, čímž umožní threadu TL co nejdříve dokončit kritickou sekci
  - Po opuštění kritické sekce threadem TL je jeho priorita vrácena na původní hodnotu
  - Teprve pak se může spustit thread TM, neboť má nižší prioritu, než byla zvýšená priorita threadu TL
- Pozor! Kdyby však thread TL byl blokován dalším threadem TLL s ještě nižší prioritou, Win CE prioritu threadu TLL už nezvýší!
- Uvedené řešení je lepší než náhodné zvýšení priory ve Windows NT/2000/XP, musí být však používáno opatrně!

18

Kvantum threadů

- Časové kvantum (time-slice) může být přidělováno jednotlivým threadům
- Před Windows CE 3 bylo přidělované časové kvantum vždy 25 ms.
- Nyní s ním lze pracovat pomocí nových Win32 API funkcí:
  - `CeSetThreadQuantum()`
  - `CeGetThreadQuantum()`
- Časové kvantum se zadává v milisekundách.
  - Implicitní hodnota je 100 ms.
  - Hodnota kvanta je volena v tzv. OEM Adaptation Layer (OAL) a může být snížena na jakoukoliv nenulovou hodnotu
  - V programu může být nastavena i hodnota 0, která říká, že thread poběží až do svého ukončení a nepustí k procesoru žádný jiný thread se stejnou prioritou
  - Příliš malá hodnota zvyšuje režii (hodně času se stráví v plánovači)
  - Příliš velká hodnota způsobuje nepřiměřené prodlevy ve spouštění threadů

19

Časovače

- Ve verzích Win CE před 3.0 byla „hrubost“ (granularity) nastavení časovače dána časovým kvantem threadu, tj. 25 ms
- Nyní je tato rozlišovací schopnost časovače oddělena od kvanta její hodnota je rovna 1 ms. Hodnota může být ještě zkrácena OEM.
- Časovače se ve Win32 API používají v různých funkcích:
  - `Sleep()` – pomocí této funkce se může thread vzdát na zadaný čas (v ms) svého kvanta.
    - V dřívějších verzích nebylo možno dodržet např. `Sleep(1)`. Mohlo být totéž co `Sleep(10)` nebo `Sleep(20)`.
  - `WaitForSingleObject()` – timeout se měří stejným časovačem jako ve funkci `Sleep()`
  - `SetTimer()` – nejméně přesný způsob měření času, vhodný zejména pro nekritické aplikace (např. s bohatým uživatelským rozhraním)

20

Správa paměti

- Může být překvapivé, že Windows CE používají stránkovanou paměť, když obvykle systémy s Win CE nemají harddisk a tedy nepodporují tzv. stránkovací souboru (**`pagefile.sys`** ve Windows NT/2000/XP)
- RAM je rozdělena na 2 části:
  - Object store – paměť pro ukládání objektů a dat.
    - Obsahuje např. souborový systém.
    - Data mohou být komprimována
    - Mohou se do ní odkládat nepotřebné stránky paměti
  - Program memory – paměť pro spouštění programů
- Nevýhoda odkládání nepotřebných stránek. V časově kritických aplikacích způsobuje natahování potřebné stránky (po výjimce page fault) nežádoucí zpoždění! Situaci lze řešit několika způsoby:
  - Odkládání stránek lze v celém systému vypnout
  - Pro zavedení kritických ovladačů do paměti lze použít funkci **`LoadDriver()`**, která je podobná **`LoadLibrary()`** až na to, že zaručuje zavedení všech stránek do paměti a jejich uzamčení

Phar Lap ETS (Embedded Tools Suite)

- Operační systém pevného reálného času dodávaný firmou Ardence (**`www.ardence.com`**)
- Základní vlastnosti:
  - Podpora 32-bitové procesorové platformy x86 firmy Intel
  - Jednoprocesový ale multithreadový OS – operační systém a aplikační program je jeden EXE soubor, který může případně volat uživatelské DLL knihovny
  - Nepotřebuje implementovat MMU (stránkovanou paměť), tím se výrazně snižuje režie systému.
  - Jeden paměťový prostor pro OS i aplikační program – může být nevýhodou
  - Nejmenší obraz ze všech RTOS – podle konfigurace desítky až stovky KB
  - Nejkratší odezva (příklad pro Pentium IV, 3 GHz)
    - Doba do zavolání hardwarové ISR < 1 mikrosekunda
    - Doba do aktivace IST z ISR = 5 mikrosekund
    - Přepnutí kontextu na thread s vyšší prioritou = 1 mikrosekunda
    - Vzdání se procerorového času funkcí **`Sleep()`** < 1 mikrosekunda
- Phar Lap ETS je využíván v zařízeních LabVIEW Real-Time Targets firmy National Intruments

Vývoj aplikací v Phar Lap ETS

- Největší výhodou je vývoj aplikací v Microsoft Visual Studiu (6.0 i .NET)
  - Vývoj mohou zahájit všichni, kdo umí programovat Windows
- Podpora rozhraní Win32 API
  - Phar Lap ETS podporuje všechny podstatné funkce z rozhraní Win32 pro vytváření procesů, threadů, synchronizační objekty (mutexy, semafory, události) !
- Interaktivní konfigurace jádra systému
  - Pro danou aplikaci vybírá tvůrce potřebné komponenty OS, které mají být přisestaveny
  - Např. ovladač obrazovky a klávesnice, souborový systém, více threadů (multithreading), TCP/IP a volba síťového adaptéru, sériové linky, zavaděč DLL, mikroweb server, FTP server, vzdálené ladění, apod.
- Využití překladačů firmy Microsoft
  - Zdrojové soubory se překládají z **`.c`**/**`.cpp`** do souborů **`.obj`** a **`.lib`** standardními překladači firmy Microsoft
  - Velkou výhodou je, že tyto překladače skutečně fungují a udržuje je větší firma než Ardence
- Linkování vlastním linkerem **`LinkLoc.exe`**
  - Linker vytvoří výsledný kód na základě konfigurace jádra OS a přeložených souborů aplikace

VxWorks

- Asi nejslavnější (a též nejdražší) RTOS vyvinutý firmou Wind River Systems (**`www.windriver.com`**)
- Používá jej např. NASA, byl na Marsu v misích PathFinder, Spirit a Opportunity
- Základní vlastnosti
  - Podpora různých platforem, např. Power PC, x86, MIPS, Sun Sparc, apod.
  - Multitasking OS. Místo pojmu thread používá pojem task
  - UNIX like OS – podporuje standardy POSIX
  - Vlastní vývojové prostředí Tornado pro Windows, Linux a UNIX
  - Konfigurovatelné součásti – desítky konfiguračních voleb