Programové řešení přechodu na zimní/letní čas

Abstrakt

Aplikační poznámka řeší automatický přechod na zimní/letní čas.

Autor: Zbyněk Říha
Dokument: ap0022_cz_02.pdf

Příloha

Obsah souboru: -

-	Není

Obsah

Historie revizí

Verze	Datum	Změny
001	3. 12. 2008	Nový dokument
002	17. 12. 2009	Změna názvu funkčního bloku

Související dokumentace

1) Nápověda k návrhovému prostředí DetStudio
   soubor: DetStudioHelp.chm

1. Automatická změna zimního/letního času

Řešení automatické změny času je individuální záležitostí, kde je možné, v několika fázích návrhu algoritmu, vybrat ze dvou různých řešení. Polovina uživatelů bude přesvědčena o správnosti jednoho řešení a druhá polovina upřednostní druhé. To je důvod, proč jsme upustili od záměru integrovat řešení problematiky zimního/letního času přímo do operačního systému nebo do funkčního modulu a rozhodli jsme se řešení nechat na autorovi aplikace, aby sám mohl rozhodnout o tom, jaké řešení mu vyhovuje. Jako návod a inspiraci nabízíme níže uvedené příklady.

Uveďme některé ze zmiňovaných problémů:

- Bez ohledu na to, jakou zvolíte metodu (jestli udržujete interní čas jako UTC a měníte offset pro výpočet lokálního času nebo udržujete interní čas jako lokální čas a udržujete si příznak, jestli je to čas zimní nebo letní) musíte si někde mimo hodinový obvod (v datech aplikace) udržovat určitý dodatečný údaj (offset nebo příznak zima/léto). To dělá problémy při zavedení aplikace (nové nebo upravené), kdy tento dodatečný údaj nemá platnou hodnotu.
- Další problém je, chceme-li zajistit (což zpravidla chceme), aby systém správně přešel na zimní/letní čas jak v případě, že byl v okamžiku změny času v chodu, tak v případě, že byl v okamžiku změny času vypnut.
- Další problém nastává na systémech, kde má probíhat automatická změna letního/zimního času a zároveň existuje možnost ručního nastavení data a času uživatelem. Pokud uživatel změní ručně datum z 15. 3. na 15. 4., má se okamžitě přepnout na letní čas tj. přičíst hodinu? Polovina lidí odpoví ano, polovina odpoví ne. A co když uživatel změní ručně čas z 1:05 na 4:05 zrovna v tu neděli, kdy probíhá změna času? Má se okamžitě přičíst/odečíst hodina, tj. nastavit čas o hodinu posunutý oproti tomu, co uživatel právě zadal? Zde patrně každý odpoví „ne“, že by se nově zadaný údaj měl respektovat a považovat za údaj v rámci nového (zimního/letního) času. Přitom rozlišit ruční změnu času od normálního plynulého chodu hodin je programově velmi obtížné, nestačí pravidelné čtení hodin a vyhodnocení okamžiku přechodu s následnou úpravou. Bylo by nutné se přímo systémově navázat na událost ruční změny času a obsluhovat ji jiným algoritmem, než změnu způsobenou chodem hodin – obejít standardní periodickou kontrolu časového údaje. K tomu ovšem v DetStudiu prakticky nejsou k dispozici potřebné prostředky.
- S ručním zadáváním času souvisí další problém – zpravidla se požaduje, aby se systém definovaně vypořádal se situací, kdy uživatel zadá neplatný časový údaj (např. 28. 3. 2008, 02:30:00 je neplatný údaj, protože ve 02:00 se změnou na letní čas hodiny posunuly na 03:00 a čas 2:30 v tomto dni jednoduše neexistuje). Zadání takovéhoto neplatného údaje je chybou uživatele, ale je pochopitelným požadavkem, aby takováto chyba nezpůsobila systému závažné problémy. Jak přesně se ovšem má reagovat, je otázkou osobního přístupu.
- Nakonec je potřeba zvážit i to, zda zákon (resp. vyhláška) definující algoritmus pro datum změny (poslední neděle v březnu / poslední neděle v říjnu) bude platit dlouhodobě, nebo zda dojde v nějaké podobě ke sjednocení Evropy s USA, kde je datum jarního přechodu druhá neděle v březnu a datum podzimního přechodu je první neděle v listopadu.

1.1. Příklad řešení 1

Nabízíme Vám následující popis algoritmu, na kterém je možné vidět jeden ze způsobů řešení.

Program vytvoříme takto:

```
//Zjištění aktuálního času v řídicím systému
GetTime RTC_time, RTC_slozky, NONE

//Převod na čas Azorských ostrovů (přechod zima/léto nastává o půlnoci)
Let UTC_time = if(@tmLetniCas, RTC_time - 10800, RTC_time - 7200)

//Převod časi z formátu DB-Net na složky
ParseTime UTC_time[0,0], NONE, NONE, UTC_slozky
```

```
//Na základě jednotlivých složek zjistíme, zda je zimní nebo letní časové období
Let @tmLeto = ((UTC_slozky[4, 0] > 3) and (UTC_slozky[4, 0] < 10) or (UTC_slozky[4, 0]
     == 3) and ((UTC_slozky[3, 0] + 7) - UTC_slozky[6, 0] > 31)) or (UTC_slozky[4, 0]
     == 10) and ((UTC_slozky[3, 0] + 7) - UTC_slozky[6, 0] <= 31)

//Po zavedení aplikace se v prvním průchodu procesem tato část neporovede
If @tmLetoInit, :NONE

     //Zjištění, zda došlo k přechodu mezi letním / zimním časem
     Let @tmZmena = @tmLeto xor @tmLetniCas
     If @tmZmena, :NONE

               //V případě, že došlo k přechodu, získáme aktuální čas a posuneme jej
               GetTime RTC_time, NONE, NONE
               Let RTC_time = if(@tmLeto, RTC_time + 3600, RTC_time - 3600)

               //posunutý čas zapíšeme zpět do řídicího systému
               SetTime RTC_time

          EndIf

EndIf

//Pomocná proměnná pro vyhodnocování přechodu mezi letním / zimním časem
Let @tmLetniCas = @tmLeto

//Pomocná proměnná pro detekci prvního průchodu procesem (po zavedení aplikace)
Let @tmLetoInit = true
```

Program je nutno vyvolávat z periodického procesu s vhodnou periodou, tak často, jak přesně chceme, aby se systém „trefil“ do okamžiku změny času z letního na zimní a zpět. Perioda ovšem nesmí být kratší než 1 sekunda (z důvodu problémů s asynchronním zápisem času do hodinového obvodu v operačním systému).

Výše uvedený algoritmus řeší změnu zimního/letního času tak, že se v hodinovém obvodu udržuje lokální (zimní/letní) čas, v databázi aplikace se udržuje příznak, jestli tento čas je letní nebo zimní (bit `@tmLetniCas`). Řeší se i situace po zavedení aplikace, kdy se aktuální čas stanice považuje za z dřívějška správně nastavený, jen se nastaví příznak, zda se jedná o čas zimní nebo letní. Otázku ruční změny data přes hranici léto/zima algoritmus řeší přístupem – upravit čas. Otázka ruční změny času přes hranici není uspokojivě vyřešena (při pokusu nastavit dne 28. 3. 2008 v 1:00 nový čas 4:00 se čas okamžitě změní na 5:00, protože okamžitě následuje automatická změna ze zimního na letní).

1.2. Příklad řešení 2

Tento příklad je modifikací příkladu řešení 1. Do programu přidáme mezi druhý a třetí řádek kódu tyto řádky:

```
If @SysStart, :NONE

Else :NONE

     Let @tmLetoInit = if((UTC_time < LastUTC_t) or (UTC_time > LastUTC_t + 2),
     bool(0), @tmLetoInit)

EndIf

Let LastUTC_t = UTC_time
```

a na konec podprogramu přidáme řádek:

```
Let @SysStart = False
```

Do procesu INIT (který případně vytvoříme) doplníme řádek:

```
Let @SysStart = True
```

Pozor
Podtržením zvýrazněná ***konstanta = 2*** *platí pro jednosekundovou periodu vyvolávání programu. Pokud program vyvoláváme s delší periodou, musí se toto číslo změnit nejméně na* ***Perioda + 1****, doporučujeme však pro jistotu volit hodnotu* ***2 × Perioda****.*

Touto úpravou se dosáhne toho, že po jakékoliv ruční změně data nebo času nedojde k následné úpravě času na zimní/letní. Pouze se podle nastaveného data a času nastaví příznak `@tmLetniCas`.

Úprava spočívá v tom, že se kontroluje, jestli se mezi dvěma vyvoláními programu nezměnil čas o více, než by odpovídalo normálnímu chodu hodin po dobu mezi vyvoláními procesu. V takovém případě se předpokládá, že došlo k ruční změně a obskočí se řádky provádějící automatickou změnu času, takže se pouze upraví hodnota příznaku `@tmLetniCas`, aby odpovídala nově zadanému datu a času.

1.3. Princip činnosti navrhovaných řešení

Do proměnné `UTC_time` se na druhém řádku programu ukládá lokální čas převedený na pásmový (=zimní) čas Azorských ostrovů, tj. čas o hodinu opožděný oproti UTC (GMT). Takto upravený čas má tu výhodu, že v něm středoevropský přechod zima/léto a zpět nastává vždy o půlnoci, takže není třeba řešit speciality různých časů v rámci kritické neděle, kdy dochází ke změně času.

S takto uvedeným časem lze zjistit, jestli spadá do zimního nebo letního období (bit `@tmLeto`) následovně:

Léto je od dubna do září, v březnu tehdy, jestliže nejbližší následující neděle je už v dubnu, v říjnu tehdy, jestliže nejbližší následující neděle spadá ještě do října.
S výjimkou okamžiku těsně po zavedení aplikace (v příkladu 2 též s výjimkou okamžiku po ruční změně času) se tento příznak porovnává s aktuální hodnotou příznaku `@tmLetniCas`, v případě neshody se nastaví příznak `@tmZmena` a čas se posune o hodinu vpřed nebo vzad.

V každém případě se bit `@tmLeto` kopíruje do bitu `@tmLetniCas`.

Proměnnou `RTC_Time` není nezbytně nutné zavádět, je možné ji na všech místech, kde se používá, nahradit proměnnou `UTC_Time`. Její použití je výhodné ve fázi testování, protože pak máme oddělené oba pracovní časy – proměnná `RTC_Time` obsahuje stále lokální čas (zimní nebo letní), proměnná `UTC_Time` obsahuje stále azorský pásmový čas – viz výše.

Pozor
Tato navrhovaná řešení platí pouze pro středoevropský čas!

1.4. Výhody a nevýhody obou řešení

- Příklad 2 lépe řeší otázku ruční změny času. Změněný čas je zachován tak, jak byl zadán, bez pokusu o automatickou úpravu na zimní nebo letní.
- Příklad 1 lépe řeší zadání neplatného časového údaje (spadajícího do vynechané hodiny při změně ze zimního na letní) – okamžitě jej přičtením či odečtením hodiny změní na platný. Příklad 2 v takové situaci nechá nastavený neplatný čas a až do jeho samovolné změny na platný (přechodem přes 03:00:00) bit `@tmLetniCas` osciluje mezi nulou a jedničkou.

Při prvním vyvolání programu, těsně po zavedení aplikace, nastává v obou příkladech jedna drobná nekorektnost, protože pro převod lokálního času na azorský (`UTC_Time`) se používá bit `@tmLetniCas`, který v tom okamžiku nemá ještě správnou hodnotu, čímž může nastat hodinový posun. Čas `UTC_Time` se potom použije pro určení rozhodného okamžiku změny času. Problémy

by to mohlo způsobit jedině v případě, že by se zaváděla nová aplikace v okamžiku méně než hodinu vzdáleném od okamžiku přechodu na zimní/letní čas. Vzhledem k malé pravděpodobnosti takovéto události nepovažujeme za nutné řešit tuto nekorektnost dalším komplikováním algoritmu. Navíc spolehlivé řešení této situace je v případě podzimní změny principiálně nemožné. V okamžiku, kdy jediná informace, která je k dispozici, je čas z hodinového obvodu, není podle čeho určit, jestli čas 31. 10. 2008, 02:30:00 je zimní nebo letní.

1.5. Funkční blok pro přechod letní/zimní čas

Pro změnu letního/zimního času byl vytvořen funkční blok s názvem RTC.sb, který je součástí instalace DetStudia. Tento je naprogramován dle příkladu řešení 2 a lze jej jakkoliv uživatelsky dále upravovat.

Pozor
Funkční blok, tak jak je napsán, musí být vložen do procesu s periodou 1 sekunda!

2. Technická podpora

Veškeré informace ohledně programového řešení přechodu na zimní/letní čas, Vám poskytne oddělení technické podpory firmy AMiT. Technickou podporu můžete kontaktovat nejlépe prostřednictvím emailu na adrese **support@amit.cz**.

3. Upozornění

AMiT spol. s r. o. poskytuje informace v tomto dokumentu, tak jak jsou, nepřejímá žádné záruky, pokud se týče obsahu tohoto dokumentu a vyhrazuje si právo měnit obsah dokumentu bez závazku tyto změny oznámit jakékoli osobě či organizaci.

Tento dokument může být kopírován a rozšiřován za následujících podmínek:

1. Celý text musí být kopírován bez úprav a se zahrnutím všech stránek.

2. Všechny kopie musí obsahovat označení autorského práva společnosti AMiT, spol. s r. o. a veškerá další upozornění v dokumentu uvedená.

3. Tento dokument nesmí být distribuován za účelem dosažení zisku.

V publikaci použité názvy produktů, firem apod. mohou být ochrannými známkami nebo registrovanými ochrannými známkami příslušných vlastníků.