AUTOMA

časopis pro automatizační techniku

www.automa.cz
Cena 52 Kč

2 ÚNOR 2013

9 771210 959006 02

Plánování úloh v systémech RT – V: zvyšování provozuschopnosti systémů

Tento, závěrečný článek seriálu bude věnován přehledu základních pojmů, jakož i popisu a nástinu řešení problémů souvisejících s provozuschopností tzv. systémů reálného času (*real-time*, RT, systémy RT). *Provozuschopností* (anglicky *dependability*), nebo také *spolehlivostí v širším smyslu*, se v tomto článku rozumí schopnost systému RT plnit funkce v souladu s podmínkami vymezenými jeho specifikací a za těchto podmínek.

Provozuschopnost jakéhokoliv technického systému závisí na mnoha činitelích, které ji ovlivňují – z nejvýznamnějších zmiňme např. *bezporuchovost* (*reliability*), *pohotovost*/*dostupnost* (*availability*), *udržovatelnost* (*maintainability*) či *bezpečnost* ve smyslu dopadu možných následků činnosti systému (*safety*) nebo ve smyslu schopnosti zaručit důvěrnost informací (*security*).

Pro kvantifikaci těchto činitelů, a tedy i celkové úrovně provozuschopnosti, existuje množství ukazatelů, např. *střední doba do poruchy* (*Mean Time To Failure* – MTTF), *střední doba provozu mezi poruchami* (*Mean Time Between Failures* – MTBF), *střední doba do opravy*/*obnovy* (*Mean Time To Repair* – MTTR) atd., jejichž popisem se však článek zabývat nebude.

Existuje také množství technik umožňujících zajistit požadovanou úroveň provozuschopnosti systému pomocí odlišných prostředků použitelných na odlišných úrovních popisu a realizace systému. Účelem tohoto článku není vyčerpávajícím způsobem pokrýt obecnou problematiku zajištění provozuschopnosti systémů, ale představit typické problémy, jejich příčiny a vybrané postupy řešení související s konstrukcí systémů RT řízených při použití operačních systémů reálného času (RTOS) s ohledem na dosažení bezporuchového provozu.

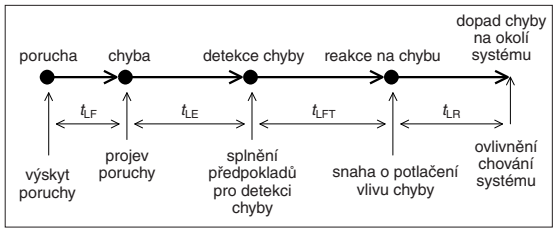
Základní pojmy

Dříve, než budou představeny konkrétní techniky používané k zajištění provozuschopnosti systémů RT, je třeba zmínit několik důležitých pojmů souvisejících s poruchami (*obr. 1*). Pro systémy RT jsou klíčová zpoždění (latence) související s těmito pojmy, jelikož obecně zvětšují pravděpodobnost nedodržení časových omezení úloh RT.

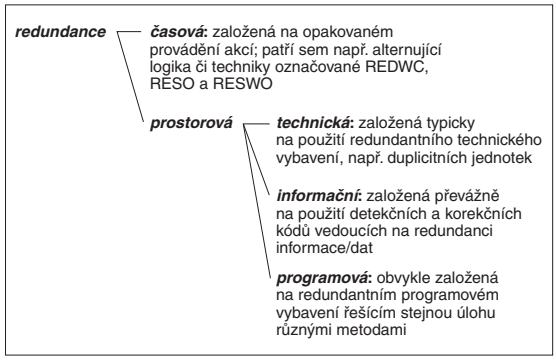
Porucha

Pojmem *porucha* (*fault*) je obvykle označována neplánovaná fyzická změna v původní struktuře obvodu, složení látky apod. Poruchy lze klasifikovat podle mnoha kritérií – např. podle doby trvání se rozlišují tyto poruchy:

– *trvalé* (*permanent*), které jsou typicky nevratného charakteru (např. zkrat) a jsou v systému přítomné, dokud nejsou opraveny nebo není vyměněna ta část systému, v níž se porucha vyskytuje,
– *přechodné* (*transient*), vyznačující se omezenou dobou trvání (např. změna stavu klopného obvodu v důsledku působení vnějšího elektromagnetického pole); jakmile pomine příčina jejího vzniku, přechodná porucha odezní,
– *občasné* (*intermittent*), projevující se obdobně jako poruchy přechodné, avšak s tím rozdílem, že jde o opakovaný a za běhu systému obtížně odstranitelný jev, obvykle důsledek nahodilé nesprávné činnosti hardwaru nebo chyby při návrhu (např. uvolněný kontakt spínače, popř. neinicializovaný ukazatel apod.).



Obr. 1. Ilustrace k možnému dopadu výskytu poruchy na chování systému



Obr. 2. Základní klasifikace metod redundance používaných ke zvýšení spolehlivosti (REDWC – Recomputing with Duplication with Comparison, RESO – Recomputing with Shifted Operands, RESWO – Recomputing with Swapped Operands)

Poruchy lze modelovat různými tzv. *modely poruch* – např. model poruch typu trvalá nula/jedna, zpoždění apod. Porucha může být skrytá, jelikož např. zasáhla podsystém toho času nevyužívaný aktuální verzí systému RT; v takovém případě je funkce systému v praxi nedotčena a porucha zůstává neodhalena.

Chyba

Není-li skrytá, porucha (např. zkrat na úrovni polovodičových vrstev v obvodu velmi vysoké integrace) se typicky projeví tzv. *chybou* (*error*), např. změnou hodnoty bitu v datech. Mezi vznikem poruchy a jejím projevem ve formě chyby však uplyne určitá doba (t_{LF} v *obr. 1*) potřebná k propagaci vlivu poruchy mezi souvisejícími částmi systému. Navíc, vznik chyby je nutným předpokladem pro její následnou detekci např. některým z bezpečnostních kódů.

Z uvedeného vztahu mezi poruchou a chybou rovněž plyne, že z hlediska chování systémů RT je ideální latence související s poruchami eliminovat či alespoň minimalizovat. Pouze tak lze eliminovat či minimalizovat i vliv poruch na chod systému RT. Tento způsob však vyžaduje přítomnost detekčních a korekčních mechanismů již na úrovni technického vybavení (hardware) systému – existují např. diagnostické detekční/lokalizační testy schopné detekovat, či dokonce lokalizovat poruchu z daného modelu poruch hardwaru ještě předtím, než je propagována do vyšších vrstev systému. V dané souvislosti je vhodné zmínit ukazatel *pokrytí poruch* (*fault coverage*), udávající procento poruch, které může být daným testem odhaleno. Nejsou-li potřebné mechanismy v hardwaru zahrnuty, lze mírnit dopad poruchy až poté, co se popř. projeví chybou.

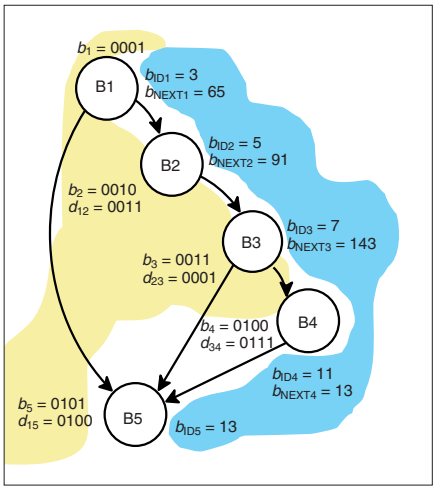
Zotavení se z chyby

Jsou-li splněny předpoklady pro detekci chyby – např. je-li příslušná část systému vybavena vhodným detekčním mechanismem, může být chyba (po uplynutí zpoždění t_{LE} spojeného s provedením mechanismu) detekována a může být proveden pokus o *zotavení se* (*obnovu*) z chyby – viz latence t_{LFT} v *obr. 1*.

Selhání

Nezdaří-li se obnova z chyby, může dojít k selhání neboli *celkové poruše* (*failure*), kdy systém přestane plnit funkci, pro kterou byl

navržen (viz latence t_{LR} v *obr. 1*). Má-li selhání vážné následky pro okolí systému, hovoří se o *havárii* či *neštěstí* (*accident*). Mnoho zejména bezpečnostně kritických systémů nebo alespoň jejich částí je proto navrženo tak, aby svou konstrukcí selhání zamezily např. zastavením veškeré své činnosti (*fail-stop*), přechodem do bezpečného stavu (*fail-safe*) či řízeným omezením svých činností (*fail-operational*).



Obr. 3. Graf programu pro ilustraci principu technik ECCA (modré pozadí) a CFCSS (žluté pozadí) kontroly toku řízení

Prostředky ke snížení poruchovosti

Ke snížení poruchovosti systému lze použít mnoho různých prostředků, které je možné rozdělit např. do těchto skupin:

– prostředky k *odstranění* poruch (*fault removal*), typicky používané v etapách formální verifikace či ověřování systému,
– prostředky k *zamezení vzniku* poruch (*fault avoidance*), obvykle realizované na úrovni hardwaru v etapě návrhu systému,
– prostředky k *predikci* poruch (*fault prediction*), založené na analýze historie poruch získané na základě provozu systému v reálném prostředí či ze statistických dat nad simulačním modelem systému,
– prostředky k dosažení *odolnosti* proti poruchám (*fault tolerance*), schopné zajistit chod systému navzdory výskytu poruch v systému.

Podle použité metody redundance lze prostředky dále rozčlenit na prostředky založené na redundanci *prostorové* (*spatial*) či *časové* (*time*), podle *obr. 2*, a podle způsobu reakce na chybu na prostředky s *dopřednou* (*forward*) či *zpětnou* (*backward*) obnovou běhu (*recovery*) atd.

S ohledem na charakter předchozích dílů tohoto seriálu budou v následujícím textu stručně představeny pouze principy vybraných prostředků zotavení se z poruch spadajících do kategorie odolnosti proti poruchám programového vybavení. Z pohledu použité metody redundance bude text omezen na prostředky realizovatelné na úrovni jádra RTOS a vyšší, tj. zejména na prostředky založené na redundanci časové a programové (*obr. 2*).

Kontrola toku řízení

Jako první si představme techniku (prostředek) umožňující detekovat chybu v toku řízení programu. Tato chyba se typicky projevuje tím, že posloupnost instrukcí prováděných procesorem je jiná, než byla navržena programátorem, např. začne být prováděn jiný podprogram, než který byl volán.

Princip této techniky lze shrnout následovně. Každý program P je nejprve rozčleněn na *bloky* B*i*, z nichž každý je tvořen nejdelší souvislou posloupností neskokových instrukcí. Na P lze pohlížet jako na orientovaný graf, tzv. *graf programu*, $G_P = (V, E)$, jehož množina uzlů V obsahuje všechny bloky z P a jehož množina hran E obsahuje všechny povolené přechody mezi bloky obsaženými v P. Cílem technik z této kategorie je ověřovat, zda prováděný přechod mezi bloky je povolen, nebo nikoliv. V případě, že povolen není, může být tato chyba indikována nadřazené jednotce, která následně rozhodne o způsobu reakce na ni.

Pro ilustraci nejprve zmiňme princip techniky označované ECCA (*Enhanced Control-Flow Checking Using Assertions*). Tato technika přiřazuje každému bloku B*i* jednak identifikátor b_{IDi}, reprezentovaný prvočíslem různým od 2, a jednak hodnotu b_{NEXTi}, rovnou součinu hodnot identifikátorů povolených následovníků bloku B*i* v daném G_P. Pro účely detekce chybného toku řízení je dále deklarována globální proměnná G_{ID}, určená k ukládání výsledků speciálních operací SET (TEST) prováděných na začátku (konci) každého bloku B*i* s použitím vztahů

$$G_{ID} = \frac{b_{IDi}}{(G_{ID} \bmod b_{IDi})(G_{ID} \bmod 2)} \qquad (1)$$

při operaci SET, popř.

$$G_{ID} = b_{NEXTi} + \mathrm{diff}(G_{ID} - b_{IDi}) \qquad (2)$$

kde „diff“ je funkce vracející pro nulový rozdíl hodnotu 0 a jinak hodnotu 1, při operaci TEST. Chybný tok řízení může být detekován každou z uvedených operací.

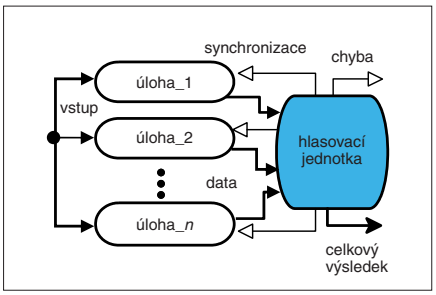
Při použití operace SET vede chyba na dělení nulou, která se objeví ve jmenovateli v jednom z těchto případů:

– buď B*i* není platným následovníkem předcházejícího bloku, jehož hodnota je uložena v G_{ID}, a výsledkem výpočtu (G_{ID} mod b_{IDi}) tedy bude hodnota 1, která se po inverzi změní na 0,
– nebo je G_{ID} sudé číslo, a tedy výsledek (G_{ID} mod 2) je roven 0.

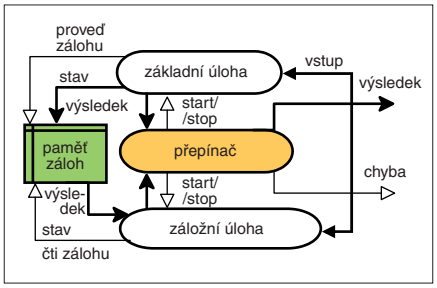
Při provádění operace TEST se do G_{ID} uloží hodnota b_{NEXTi} zvětšená o hodnotu rozdílu ($G_{ID} - b_{IDi}$), který je nulový, končí-li program v bloku, který začal poslední operací SET. V opačném případě je tento rozdíl sudý, což při následné operaci SET povede na dělení

ní nulou. Ilustrace k technice ECCA pro program složený z bloků B1 až B5 je na *obr. 3*, obsahujícím graf programu a hodnoty b_{ID}, b_{NEXT} pro jeho uzly.

Na obdobném principu je založena technika CFCSS (*Control-Flow Checking by Software Signatures*). Její předností je, že pro identifikaci bloku B*i* nevyžaduje prvočíslo a pro detekci správnosti toku nepoužívá dělení – požadavky na paměť a výpočetní výkon jsou tedy menší než u ECCA. Pro detekci chybného kroku řízení pak CFCSS namísto použití operací SET, TEST přiřazuje každému bloku B*i* vedle identifikátoru b_i hodnotu d_{ji} rovnou rozdílu $b_j \oplus b_i$, kde b_j je identifikátor bloku B*j*, který je platným předchůdcem bloku B*i*, kde $\oplus$ představuje operaci XOR. Při vstupu do nového bloku B*i* je nejprve testem ověřeno, zda z předchozího bloku B*j*, jehož identifikátor b_j je uložen v globální proměnné G_{ID}, se platně pokračuje v cílovém bloku. Test je jednoduchý, spočívající v porovnání hodnot $G_{ID} \oplus d_{ji}$ a b_i. Je-li tok řízení správný, hodnoty jsou si rovné. Technika CFCSS nedostačuje v situaci, kdy jednomu cílovému bloku předchází několik bloků zdrojových. Za této situace mohou některé chyby v toku řízení zůstat nedetekovány.



Obr. 4. Ilustrace k N-verznímu programování



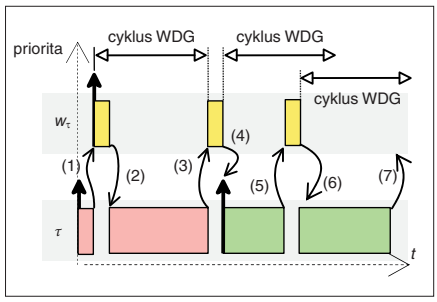
Obr. 5. Ilustrace k technikám využívajícím paměť záloh

N-verzní programování

Odolnost proti poruchám lze také zajistit zajištěním souběžného či současného běhu různých realizací úloh řešících zadaný problém – hovoří se pak o *N-verzním* či *N-násobném programování*, inspirovaném technickou N-modulovou redundancí (NMR).

Výstup každé z *n* realizací úloh je vyhodnocen *hlasovací jednotkou* (*voter*), rozhodující o tom, který z potencionálně odlišných výsledků bude vzat jako správný. Mechanismů hlasování existuje mnoho, např. *majoritní*, indikující za správný ten z výsledků, který produkovala nadpoloviční většina úloh, či *mediánový*, indikující ten z výsledků, který je mediánem hodnot na výstupech úloh.

Nicméně platí, že hlasovací mechanismy bývají navrhovány s ohledem na důraz kladený na konkrétní složky spolehlivosti – je-li důležitá bezporuchovost, musí být vybraný celkový výsledek s velkou pravděpodobností skutečně správný; je-li důležitá dostupnost, hlasovací jednotka produkuje výsledek i tehdy, nemůže-li o správnosti výsledku rozhodnout; je-li důležitá bezpečnost, je nutné vhodně klasifikovat a maskovat nesprávné výsledky atd. Nedokáže-li hlasovací jednotka poskytnout nadřazené vrstvě výsledek, signalizuje namísto něj chybu (*obr. 4*).



Obr. 6. Ilustrace k principu činnosti diagnostické úlohy w_τ

Zotavení při použití záloh

Další mechanismus ke zvýšení spolehlivosti programového vybavení je založen na využití tzv. paměti záloh, využívané k ukládání bezchybného stavu (bodu obnovy) rozpracované úlohy. Je-li za běhu úlohy detekována chyba, je z paměti záloh obnoven posledně uložený stav úlohy a od něj začne být úloha prováděna znovu. Rozložení záloh v čase může být různé – např. periodické či náhodné.

Vedle základní úlohy provádějící zálohy může systém obsahovat i jednu nebo několik tzv. záložních úloh schopných načíst data ze zálohy a běžet s nimi namísto úlohy základní – ta může být zatím odstavena a poté např. testována na bezchybnost kódu a při nalezení chyby nahrazena výchozí verzí. Selžou-li jak základní, tak i záložní úlohy, je nadřazené jednotce signalizována chyba, tj. neschopnost danou funkci provést. Varianta této techniky založená na jedné základní a jedné záložní úloze je ilustrována na *obr. 5*.

Diagnostické úlohy

Poslední skupina technik používaných ke zvýšení odolnosti proti poruchám na úrovni programového vybavení, která bude v tomto článku představena, obsahuje techniky založené na využití tzv. diagnostických úloh. Inspirací pro vznik těchto technik byla existence periferie typu *watchdog* (WDG) na čipech mikrořadičů. Princip této periferie je jednoduchý – funkce watchdog čeká po předem stanovenou dobu (perioda cyklu WDG) na přijetí signálu. Přijde-li během této doby signál, je zahájen další cyklus WDG. Jinak funkce watchdog vyvolá nové spuštění (*reset*) mikrořadiče za účelem rychlé obnovy činnosti systému nacházejícího se např. ve stavu zacyklení či uváznutí. Výskyt nežádoucích stavů je pak vyvozen z neschopnosti systému včas signalizovat svoji správnou funkci periferii typu watchdog.

Analogií periferie typu watchdog v oblasti programového vybavení mohou být např. tzv. diagnostické úlohy (úlohy WDG). Princip spočívá v tom, že každé RT úloze τ je přiřazena periodická úloha WDG s prioritou významnější než τ (úloha w_τ). Úloha w_τ musí být spuštěna co nejdříve po spuštění τ – viz *obr. 6*, vazba (1) – za účelem včasné detekce chyby v τ a případné aktivace mechanismu zotavení se z této chyby. Obdrží-li úloha w_τ do konce své periody (cyklu WDG) signál od úlohy τ, otestuje stav úlohy τ na přítomnost chyb (každá úloha ukládá svůj stav do paměti obnovy obsahující také posledně známý bezchybný stav). Je-li detekována chyba, popř. není-li úloha w_τ signalizována včas – vazba (3), je úlohou w_τ spuštěn mechanismus obnovy: úloha τ může být např. spuštěna znovu – vazba (4) – s nadějí, že v následné instanci bude dokončena bezchybně – viz vazby (4) až (7). S dokončením úlohy τ dojde i k zastavení činnosti jí příslušející úlohy w_τ – viz *obr. 6*, vazby (4) až (7). Náznak možné realizace této techniky je uveden na *obr. 7* – k odměřování cyklu WDG je zde použit nastavitelný časový limit čekání na příchod zprávy do schránky zpráv.

Správnou činnost diagnostických úloh lze sledovat zavedením nadřazené (systémové) diagnostické úlohy, jejímž úkolem je kont-

```
01 static void task_n_body (void *p_arg)
02 {
03    OSTaskCreate(task_n_WDG,/* WDG task creation */
04      (void *)0,
05      (OS_STK *)&task_n_wdg_stk[TASK_WDG_STK_SIZE-1],
06      TASK_n_WATCHDOG_PRIO);
07
08    for(;;) /* -- code of the task -- */
09    {
10      /* init data */
11      /* perform function */
12      /* create check & restoration points */
13      /* produce output */
14      OSTimeDly(TASK_n_PERIOD);/* wait for new period */
15    }
16
17    OSTaskDel(TASK_n_WATCHDOG_PRIO);  /* WDG deletion */
18 }
```

```
01 static void task_n_WDG (void * p_arg)
02 {
03    INT8U err;
04    void *msg;
05
06    for(;;)
07    {
08      msg = OSMboxPend(msg_WDG_n, WDG_n_TIMEOUT, &err);
09      if(msg != (void *)0)/* -- WDG reset on-time -- */
10      {   /* check for omission and value faults
11      }       then reset SYS_WDG */
12      else
13      { /* -- WDG timeout over -- */
14        if (msg != (void *)0)
15        { /* initiate recovery mechanism */ }
16      }
17    }
18 }
```

Obr. 7. Ilustrace k realizaci diagnostické úlohy prostředky RTOS µC/OS-II

rolovat zejména „živost“ podřazených diagnostických úloh [1]. Nepracuje-li některá z podřazených diagnostických úloh správně či vzroste-li počet nových spuštění jí diagnostikovaných úloh nad předem danou mez, nadřazená diagnostická úloha obě tyto úlohy spustí znovu. Vzhledem ke své klíčové roli však nadřazená úloha (jednotka) musí být také dostatečně robustní ve smyslu zvýšené odolnosti proti poruchám.

Shrnutí

V článku je nastíněna problematika návrhu spolehlivých systémů RT s důrazem na možná řešení realizovatelná prostředky programového vybavení a jader RTOS. Snahou bylo ukázat základní principy a problémy související s použitím těchto prostředků, zejména jejich vlastnost reagovat na případnou poruchu s větším zpožděním než prostředky realizované ve vrstvách bližších hardwaru. Smysluplným a účinným se jeví použít tyto prostředky vyšších úrovní jako základní či doplňkové techniky např. v oblasti odolnosti proti přechodným poruchám či chybám při navrhování a implementaci programového vybavení, které lze v nižších vrstvách jen obtížně detekovat a opravit. Mnohé z dalších technik však ani zmíněny nebyly – patří mezi ně např. techniky izolace chyb, které se snaží zamezit šíření chyb vhodnou modularizací a dekompozicí systému. Z této kategorie zmiňme alespoň techniku *uzavírání podsystému*, založenou na myšlence autorizovat každou akci před tím, než bude prováděna, popř. techniku *atomických akcí*, omezující interakci uvnitř systému v daném časovém intervalu pouze na podmnožinu prvků; každá z takto omezených akcí buď skončí bezchybně, nebo chybou – jelikož může ovlivnit pouze úzce vymezené prvky, jsou okruh jejího šíření i prostor zotavení značně omezeny [2].

Závěr seriálu

Že návrh systémů RT je složitý, ovšem plyne i z předchozích článků [3] až [6] tohoto seriálu o plánování úloh v systémech RT. Jelikož tento článek je současně posledním článkem seriálu, na závěr stručně připomeňme a shrňme články jemu předcházející.

První článek [3] byl zaměřen na přehled problémů spojených s plánováním množin závislých úloh RT – byly v něm představeny základní typy závislostí mezi úlohami (časová, prostorová) a nastíněn vliv těchto závislostí na konstrukci plánovače; mj. byly přestaveny protokoly NPP (*Non Preemptive Protocol*), PIP (*Priority Inheritance Protocol*) a PCP (*Priority Ceiling Protocol*) přístupu k prostředkům. Ve druhém článku [4] bylo představeno několik technik společného plánování periodických a neperiodických úloh s důrazem na principy serverů úloh. Mimo jiné byly popsány principy vyzývacího (*Polling Server* – PS), odkládacího (*Deferrable Server* – DS) a sporadického serveru (*Sporadic Server* – SS) a serveru s výměnou priorit (*Priority Exchange Server* – PE). Třetí článek seriálu [5] byl věnován problematice plánování úloh při přetížení systému. Zejména byla diskutována základní východiska z přetížení založená na řízení degradace výkonu systému a zavedení důležitosti úloh. Čtvrtý, tj. předposlední článek [6] tohoto seriálu shrnul problematiku plánování úloh RT ve víceprocesorovém prostředí a představil základní mechanismy přidělování procesorů úlohám vycházející z mechanismu RM (*Rate Monotonic Next Fit* – RMNF, *Rate Monotonic First Fit* – RMFF, *Rate Monotonic Best Fit* – RMBF).

V seriálu bylo záměrně odkazováno pouze na produkty UPPAAL, TimesTool, Cheddar, uC/OS-II, FreeRTOS, QNX atd., které jsou široké veřejnosti volně dostupné k nekomerčnímu využití za podmínek daných vlastníky příslušných práv. Účelem bylo, aby po přečtení jednotlivých článků seriálu měl každý možnost si tyto produkty zdarma vyzkoušet a snadno si s jejich pomocí ověřit platnost principů v článcích popisovaných.

Poděkování

Tento článek byl vypracován v rámci projektu Centrum excelence IT4Innovations (reg. č. CZ.1.05/1.1.00/02.0070), podporovaného Operačním programem Výzkum a vývoj pro inovace, financovaného ze strukturálních fondů EU a ze státního rozpočtu ČR, projektu Metodiky pro návrh systémů odolných proti poruchám do rekonfigurovatelných architektur - vývoj, implementace a verifikace (MSMT LD12036), projektu Výzkum informačních technologií z hlediska bezpečnosti (CEZ MSM 0021630528) a grantu BUT FIT-S-11-1.

Literatura:

[1] ČELEDA, P.: *Zvýšení spolehlivosti a diagnostika operačních systémů pracujících v reálném čase*. Disertační práce, Univerzita obrany, Brno, 2007.

[2] SLIMAŘÍK, F.: *Mechanismy zvýšení spolehlivosti vestavěných systémů pracujících v reálném čase*. Diplomová práce, FIT VUT v Brně, 2010.

[3] STRNADEL, J.: *Plánování úloh v systémech RT – I: závislé úlohy*. Automa, 2012, roč. 18, č. 10, s. 42–45, ISSN 1210-9592.

[4] STRNADEL, J.: *Plánování úloh v systémech RT – II: neperiodické úlohy*. Automa, 2012, roč. 18, č. 11, s. 44–46, ISSN 1210-9592.

[5] STRNADEL, J.: *Plánování úloh v systémech RT – III: přetížení systému*. Automa, 2012, roč. 18, č. 12, s. 44–47, ISSN 1210-9592.

[6] STRNADEL, J.: *Plánování úloh v systémech RT – IV: víceprocesorové prostředí*. Automa, 2013, roč. 19, č. 11, s. 44–46, ISSN 1210-9592.

Ing. Josef Strnadel, Ph.D.,
Centrum excelence IT4Innovations,
Fakulta informačních technologií,
Vysoké učení technické v Brně
(strnadel@fit.vutbr.cz)

ADRESÁŘ VYDAVATELSTVÍ

Ředitel: ..... Ing. Emil Širůček

Adresa: ..... Pod Vodárenskou věží 4, 182 08 Praha 8
tel.: 286 583 011-12, 266 052 804, fax: 284 683 022
e-mail: automa@fccgroup.cz, www.automa.cz

Šéfredaktor: .....Ing. Petr Bartošík
Zástupce šéfredaktora: ..... Ing. Eva Vaculíková
Redakce: .....Petr Špůr, Milena Kočišová
Speciální projekty: .....Ing. Karel Suchý, Ing. Ladislav Šmejkal, CSc.

Odborná spolupráce:
Ing. Karel Bílek, Bernecker + Rainer Industrie Elektronik, Ges. m. b. H., **Ing. Miroslav Dub, CSc.**, Sidat, spol. s r. o., **prof. RNDr. Ing. Petr Fiala, CSc., MBA**, katedra ekonometrie Vysoké školy ekonomické v Praze, **Ing. Otto Havle, CSc., MBA,** FCC průmyslové systémy, s. r. o., **doc. Ing. Petr Horáček, CSc.**, FEL ČVUT v Praze, **Ing. Zdeněk Hurák, Ph.D.**, katedra řídicí techniky FEL ČVUT v Praze, **doc. Ing. Josef Janeček, CSc.**, katedra řídicí techniky Fakulty mechatroniky a mezioborových studií TU Liberec, **doc. Ing. Karel Kadlec, CSc.**, ústav fyziky a měřicí techniky, Fakulta chemicko-inženýrská, Vysoká škola chemicko-technologická v Praze, **Ing. Petr Kašík,** Amit, spol. s r. o., **doc. Dr. Ing. Vladimír Kebo**, institut ekonomiky a systémů řízení, Hornicko-geologická fakulta, VŠB-TUO Ostrava, **Ing. Marie Martinásková, Ph.D.**, ústav přístrojové techniky, Fakulta strojní ČVUT v Praze, **prof. Ing. Vladimír Mařík, DrSc.**, katedra kybernetiky FEL ČVUT v Praze, **doc. Ing. Pavel Nahodil, CSc.**, katedra kybernetiky FEL ČVUT v Praze, **prof. Ing. Miloš Schlegel, CSc.,** katedra kybernetiky FAV ZČU v Plzni, **prof. Ing. Vilém Srovnal, CSc.,** katedra kybernetiky a biomedicínského inženýrství, Fakulta elektrotechniky a informatiky VŠB-TUO Ostrava, **prof. Ing. Bohumil Šulc, CSc.,** ústav přístrojové techniky, Fakulta strojní ČVUT v Praze, **prof. Ing. Vladimír Vašek, CSc.,** Fakulta aplikované informatiky, Univerzita Tomáše Bati ve Zlíně, **prof. Ing. Petr Vavřín, DrSc.**, ústav automatizace a měření FEKT VUT Brno, **prof. Ing. František Zezulka, CSc.**, ústav automatizace a měřicí techniky FEKT VUT Brno
Inzerce:.....Ladislava Hošmánková
Administrace časopisu: ..... Ing. Petra Huňová
Sazba a grafická úprava: ..... Dana Pecháčková, Tomáš Petr

SEZNAM INZERENTŮ

FROM CONTENTS

Automa. Vydává firma FCC Public s. r. o. Přetisk je dovolen jen se svolením redakce a s uvedením pramene. Za případné závazky ke třetím stranám ručí autor. Názory autorů nemusejí být shodné se stanoviskem redakce. Vydavatel nezodpovídá za pravdivost údajů uvedených v inzerci a PR příspěvcích. Pro předplatitele v České republice provádí distribuci v zastoupení vydavatele společnost Send Předplatné, Ve Žlíbku 1800/77, 193 00 Praha 9 Horní Počernice; příjem objednávek a reklamace: tel.: 225 985 225, fax: 225 341 425, send@send.cz, www.send.cz. Pro Slovenskou republiku: Magnet Press Slovakia s. r. o., Teslova 12, P. O. Box 169, 830 00 Bratislava, tel.: +421 244 454 559 (předplatné), +421 244 454 628 (sekretariát), predplatne@press.sk, Elez, Zlatovská 27, 911 05 Trenčín, tel.: +421 326 527 672, fax: +421 327 436 536, elez@elez.sk, Mediaprint-Kapa Pressegrosso, a. s., odd. inej formy predaja, Vajnorská 137, P. O. Box 183, 830 00 Bratislava 3, tel.: +421 244 458 821, +421 244 458 816, +421 244 442 773, fax: +421 244 458 819, e-mail: predplatne@abompka-pa.sk a Slovenská pošta, SPT, Nám. slobody 27, 810 05 Bratislava, objednávky prijíma každá pošta a poštový doručovateľ. Objednávky do zahraničí vyřizuje Mediaservis s. r. o., Paceřická 1, 193 00 Praha – Horní Počernice, tel.: 271 199 250, kauerova@mediaservis.cz. Veškeré objednávky přijímá také redakce, která zprostředkuje i případné reklamace. Vychází 12× ročně. Tiskne Kavka Print, a. s., Ke Zdibsku 620, 250 67 Klecany, tel.: 317 070 745. Do tisku předáno 11. 2. 2013, vyšlo 14. 2. 2013. Cena časopisu: 52 Kč (dvojčíslo 104 Kč).