AUTOMA

časopis pro automatizační techniku

www.automa.cz
Cena 52 Kč

10 ŘÍJEN 2012

- **Indukční průtokoměry – princip, vlastnosti a použití**
- **Distribuce ropy ropovody Družba a IKL**
- **Norma API 2350 – ochrana před přeplněním v novém**
- **Plánování úloh v systémech reálného času**
- **Proč používat Windows Embedded**

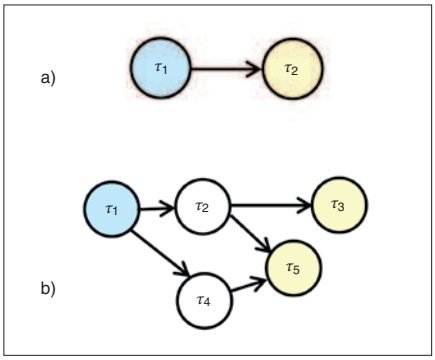
9 771210 959006 10

Plánování úloh v systémech RT – I: závislé úlohy

V seriálu článků *Návrh časově kritických systémů I* až *IV* [4] až [7] byl představen model úloh tzv. reálného času (*real-time*, RT, úlohy RT) [5] a základní preemptivní mechanismy (RM, DM, EDF, LLF) přiřazování priorit a plánování množin úloh RT [6]. Bylo předpokládáno, že úlohy jsou periodické a nezávislé a jsou bezchybně prováděny v rámci jednoprocesorového systému, který je vždy provozuschopný a má dostatečný výkon ke včasnému provedení všech úloh zahrnutých v systému. Nyní začínající seriál *Plánování úloh v systémech RT* navazuje na uvedená publikovaná témata a je věnován mechanismům, které jsou použitelné v situacích, kdy již zmíněné předpoklady splněny nejsou. I nadále však bude pojednáváno o preemptivních variantách mechanismů.

Tento první článek nového seriálu je věnován principům přiřazování priorit závislým úlohám RT, tj. úlohám, mezi kterými existují závislosti plynoucí ze specifikace systému [4]. Oproti nezávislým úlohám je tudíž za běhu systému třeba dodržet nejen časová omezení kladená na odezvy úloh, ale navíc při snahách o jejich dodržování i reflektovat dané závislosti a řešit problémy z toho vyplývající.

Mezi úlohami lze rozlišit dva typy závislostí – *časovou* a *prostorovou*. Časová závislost (někdy také označovaná jako precedenční) mezi úlohami existuje tehdy, plyne-li ze specifikace požadavek přednostního provedení jedné z úloh před jinými. Prostorová závislost mezi úlohami plyne z požadavku úloh přistupovat ke stejnému (tj. těmito úlohami sdílenému) prostředku (typicky zdroji/poskytovateli určité služby), mnohdy vyžadujícímu vzájemně výlučný přístup v tom smyslu, že daný prostředek může v daném čase využívat nejvýše jedna úloha. Tyto závislosti mohou být kombinovány, čímž se situace dále

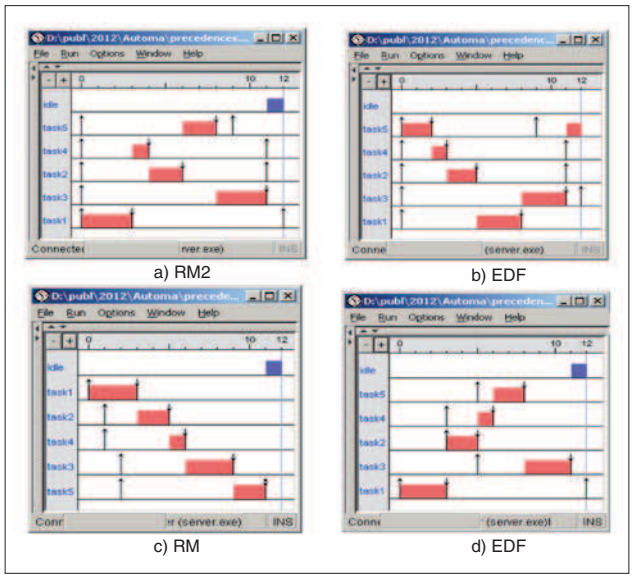


Obr. 1. Ilustrace k precedenčnímu grafu

komplikuje. V dalším textu budou nastíněny principy přiřazování priorit pro každý ze zmíněných typů závislostí.

Časová závislost mezi úlohami

Časové závislosti (angl. *precedence constraints*) mezi úlohami jsou typicky vyjadřovány a zadávány ve formě tzv. *precedenčního grafu* (*precedence graph*), což je orientovaný graf $G_P = (V, E)$, jehož uzly z množiny V představují úlohy a jehož orientované hrany z množiny $E \subseteq V \times V$ svou orientací vyjadřují časové závislosti mezi úlohami [3]. Každá časová závislost tvaru „úloha τ_i předchází úloha τ_j“, obvykle značenou $\tau_i \rightarrow \tau_j$, a je v precedenčním grafu reprezentována pomocí uzlů τ_i, τ_j a orientované hrany $(\tau_i, \tau_j) \in E$ vedoucí z τ_i do τ_j. Tento typ závislosti lze považovat za základní prostředek zajištění synchronizace akcí prováděných různými úlohami. Příkladem může být úloha τ_1 zajišťující periodické vzorkování určené veličiny (např. teploty) a úloha τ_2 periodicky produkující průměr ze čtyř posledních vzorků poskytnutých úlohou τ_1. Tuto závislost reprezentuje precedenční graf na *obr. 1a*.



Obr. 2. Plány RM a EDF pro množinu úloh s původními parametry (a, b) a s parametry modifikovanými (c, d) podle precedenčního grafu z obr. 1b (pozn.: vyobrazené plány jsou výstupem z nástroje Times Tool [8])

Obecně pro zajištění $\tau_i \rightarrow \tau_j$ musí platit: $r_j \geq r_i$ (tj. τ_j nesmí být vyvolána dříve než τ_i) a $P_i \geq P_j$ (tj. τ_j nesmí mít významnější prioritu než τ_i). Pro příklad z *obr. 1a* to znamená nastavit r_2 na hodnotu z intervalu $(r_1 + 3T_1, r_1 + 4T_1)$ priority P tak, aby platilo $P_1 \geq P_2$.

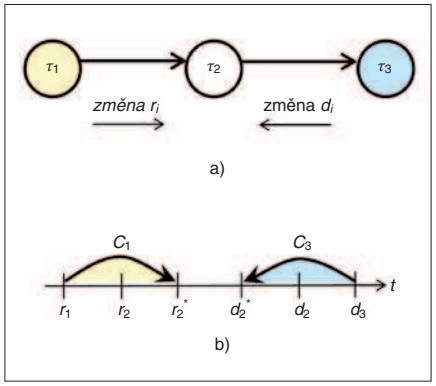
Důležité je si uvědomit, že zavedením časových závislostí již plánovač nerozhoduje o přidělení procesoru jen na základě mechanismu přiřazování priorit aplikovaném na parametry úloh, ale také na základě vztahů zaznamenaných v precedenčním grafu. V následujícím textu bude ukázáno, jak informace zaznamenaná v precedenčním grafu změní parametry úloh, a tím zajistí příslušnou časovou závislost pro daný mechanismus přiřazování priorit. Základní myšlenka již byla naznačena dříve – spočívá v zamezení spuštění úlohy před jejími předchůdci a v zamezení přerušení následovníků úlohy některým z jejích předchůdců.

V případě mechanismu RM musí být pro zajištění $\tau_i \rightarrow \tau_j$ parametry r_i, r_j úloh τ_i, $^*\tau_j$ změněny na r_i^*, r_j^* tak, aby platilo $r_j^* \geq \max(r_i, r_i^*)$ a $P_i \geq P_j$ (*tab. 1*). Zde je vhodné poznamenat, že pro úlohy se stejnou periodou tedy plyne poměrně značná volnost při volbě nejen dob vyvolání úloh (r_i), ale i priorit, které mají zajistit respektování precedenčních omezení.

Obdobným způsobem jsou priority přiřazeny i v případě mechanismu DM – zde musí, vedle podmínek pro mechanismus RM, navíc platit podmínka pro modifikace D_i^*, D_j^* parametrů D_i, D_j úloh τ_i, τ_j: $D_j^* \geq \max(D_j, D_i^*)$. Mechanismus DM totiž přiřazuje významnější prioritu úloze s menší hodnotou parametru D, tj. kratší časovou mezí odezvy.

Jelikož mechanismy RM a DM jsou statické v tom smyslu, že přiřazují priority úlohám pevně a neměnně za běhu systému pracujícího v reálném čase (systém RT), je vliv precedenčního grafu na priority omezen pouze na stanovení priorit v návrhové etapě, tj. před zahájením běhu systému RT. Situace se však komplikuje, jsou-li časové závislosti kombinovány s mechanismem dynamického přiřazování priorit, jako je např. EDF (*Earliest Deadline First*). Podle tohoto mechanismu jsou totiž úlohám přiřazovány priority nepřímo úměrně hodnotě parametru d určeného vztahem $d = D - t$, kde t je aktuální čas [5]. Hodnota d je zřejmě obecně proměnná v čase, a tedy i graf omezení musí být analyzován, a to i za běhu systému, kdykoliv je volán plánovač.

Přiřazování priorit EDF s ohledem na precedenční graf musí pro každou závislost $\tau_i \rightarrow \tau_j$ splňovat následující podmínky: $r_j^* \geq \max(r_i^* + C_i, r_j)$, tj. τ_j nesmí být vyvolána dříve, než by končila τ_i, a $d_i^* \geq \min(d_j^* - C_j, d_i)$, tj. časová mez τ_i nesmí být předsunuta před čas včasného zahájení τ_j. Při modifikaci parametrů se postupuje nejprve v protisměru orientace hran precedenčního grafu, a to od úloh



Obr. 3. Ilustrace k principu vlivu precedenčního grafu (a) na změny parametrů úloh plánovaných pomocí EDF (b)

nemajících žádné následovníky (modifikace parametru d), a poté po směru orientace hran precedenčního grafu, a to od úloh nemajících žádné předchůdce (modifikace parametru r), popř. naopak – viz *obr. 3*.

Princip zmíněné modifikace lze ilustrovat jednoduchým příkladem při použití množiny úloh z *tab. 1* a precedenčního grafu z *obr. 1b*. Změna původních parametrů začne od úloh nemajících předchůdce – v popisovaném případě od úlohy τ_1. Její parametr r_1 se změní na hodnotu $r_1^* = \max(0, r_1) = 0$. Poté se zpracovávají následovníci. Těmi jsou τ_2, τ_4 – mění se tedy r_2, r_4 následovně: $r_2^* = \max(r_1^* + C_1, r_2) = 3$, $r_4^* = \max(r_1^* + C_1, r_4) = 3$. Další změna se provede u úlohy τ_3, následovníka τ_2: $r_3^* = \max(r_2^* + C_2, r_3) = 5$. Jako poslední se změní r_5; jelikož τ_5 je následovníkem τ_2 i τ_4, hodnota bude změněna na $r_5^* = \max(\max(r_2^* + C_2, r_4^* + C_4), r_5) = 5$.

*Tab. 1. Ilustrace vlivu precedenčního grafu z obr. 1b na parametry úloh (změněné parametry jsou v pravém horním indexu označeny symbolem *)*

Úloha ($D_i = T_i$)	Původní parametry			Parametry pro RM			Parametry pro EDF			
	r_i	C_i	d_i	r_i^*	P_i	P_i^*	r_i^*	d_i^*	P_i	P_i^*
τ_1	0	3	12	0	5	10	0	5	5	10
τ_2	0	2	11	1	8	8	3	7	8	8
τ_3	0	3	12	2	5	5	5	12	5	2
τ_4	0	1	11	1	8	8	3	7	8	8
τ_5	0	2	9	2	10	5	5	9	10	5

Po vyčerpání všech následovníků se začne druhá fáze změn, tj. změny parametru d úloh. V tomto případě se začne od úloh bez následovníků, což jsou úlohy τ_3, τ_5. Hodnoty d_3, d_5 se změní na $d_3^* = \min(\infty, d_3) = 12$, $d_5^* = \min(\infty, d_5) = 9$. Poté se zpracují předchůdci úloh τ_3, τ_5, tj. úlohy τ_2 a τ_4. Protože τ_4 má jediného následovníka (τ_5), bude d_4 změněna na $d_4^* = \min(d_5^* - C_5, d_4) = 7$. Úloha τ_2 má však následovníky dva (τ_3 a τ_5), a proto bude d_2 změněna na $d_2^* = \min(\min(d_3^* - C_3, d_5^* - C_5), d_2) = 7$. Poslední úloha ($\tau_1$) má předchůdce τ_2, τ_4 a d_1, tedy bude změněna na $d_1^* = \min(\min(d_2^* - C_2, d_4^* - C_4), d_1) = 5$.

Kromě časové závislosti může mezi úlohami existovat i závislost prostorová; zbylá část článku bude tedy věnována právě jí.

Prostorová závislost mezi úlohami

Prostorová závislost mezi úlohami vzniká, požadují-li úlohy přístup ke stejnému prostředku (*resource*). S tímto typem závislosti je spojeno množství obecně známých problémů, které je třeba řešit, nicméně na jejich detailní popis není v tomto článku prostor, a proto zde bude uveden pouze přehled nejvýznamnějších z nich: *hladovění* (*starvation*), *blokování* (*blocking*), *inverze priorit* (*priority inversion*) a *uváznutí* (*deadlock*). V článku budou vynechány typické metody řešení těchto problémů (jakými jsou např. bankéřův algoritmus) v konvenčních operačních systémech a místo toho bude pozornost zaměřena na mechanismy typicky implementované v operačních systémech reálného času (RTOS) [1]. Pro jednoduchost bude opět předpokládáno jednoprocesorové prostředí.

Hladovění

Hladovění úloh nepředstavuje na třídě systémů RT závažný problém – naopak, lze říci, že patří k základní vlastnosti systémů RT v tom smyslu, že s klesající významností priority doba čekání úloh na přidělení procesoru roste. Toto čekání je zcela přirozené a plyne ze samotné podstaty systémů RT, pro něž je klíčové dodržet časová omezení, byť za cenu hladovění některých úloh.

Jelikož zbylé ze zmíněných problémů již mohou vést ke zpoždění odezev úloh a následně k nedodržení časových omezení, musí být jejich řešení při vypracovávání návrhu a realizaci systémů RT věnována patřičná pozornost. V následujícím textu bude ukázána podstata těchto problémů, jaké prostředky prevence těchto problémů existují a jaké jsou jejich vlastnosti. Pozornost bude zaměřena pouze na přehled tzv. *protokolů přístupu k prostředkům* (*resource access protocols*). Ty je možné rozčlenit na protokoly pro jedno- či víceprocesorové prostředí, na protokoly pracující se statickými či dynamickými prioritami atd. Přehled bude omezen na jednoprocesorové prostředí a statické priority.

Blokování

Jako první z problémů s dopadem na dodržování časových omezení zmiňme blokování. To lze neformálně definovat jako čekání úlohy s vysokou prioritou na uvolnění prostředku momentálně užívaného úlohou s nízkou prioritou. Toto čekání je jistě nežádoucí – prioritní úloha se totiž chystala (přednostně) provést další část svého kódu, ale vlivem vnějších okolností (tj. nedostupnosti zdroje) musela být pozastavena až do času uvolnění tohoto zdroje, což prodlužuje dobu provádění i odezvy prioritní úlohy. Blokování je ilustrováno na *obr. 4*, kde úloha τ_n s nejméně významnou prioritou začne jako první z úloh využívat prostředek (nazvěme jej např. R). Čas žádosti úlohy o přístup k R je v obrázku vyznačen symbolem $\downarrow_R$, čas uvolnění R symbolem $\uparrow_R$ a doba přístupu k R červeným obdélníkem. V čase t_1 je spuštěna úloha τ_1 s nejvýznamnější prioritou. Ta vyvolá, před zahájením svého běhu, přerušení provádění (*preempci*) úlohy τ_n. V čase t_2 však τ_1 požaduje přístup k (dosud neuvolněnému) R. Tento přístup není umožněn, procesor je τ_1 odebrán a předán úloze τ_n, aby tato dokončila R, uvolnila jej a τ_1 poté mohla pokračovat v běhu (čas t_4). Popsaná situace představuje tzv. *přímé* (*direct*) blokování úlohy τ_1 úlohou τ_n, způsobené nedostupností prostředku (v *obr. 4* vyznačené jako B_D). Mimoto existují i další typy blokování, např. *nepřímé* (*indirect*, v *obr. 4* vyznačené jako B_I), obvykle zapříčiněné dočasným propůjčením vysoké priority úloze s nízkou prioritou pro minimalizaci doby B_D, či *řetězené* (*chained*) blokování, jež může nastat, existuje-li několik úloh sdílejících několik prostředků – úloha přistupující k n prostředkům pak může být blokována až n-krát.

Tab. 2. Shrnutí vlastností protokolů NPP, PIP a PCP

Vlastnost	Protokol		
	NPP	PIP	PCP
Zamezuje nežádoucímu blokování?	ne	ano	ano
Počet blokování při sdílení k prostředků	1	k	1
Zamezuje inverzi priorit?	ano	ano	ano
Zamezuje uváznutí?	ano	ne	ano
Realizační požadavky	malé	malé	velké

Poznamenejme, že k prostředkům se vzájemně výlučným přístupem se obvykle přistupuje v rámci tzv. *kritických sekcí* (KS), jejichž provádění je atomické v tom smyslu, že v KS se může nacházet nejvýše jedna úloha. Slouží-li KS k zajištění výlučného přístupu k prostředku R, lze také hovořit o KS prostředku R (KS_R). Je-li v KS_R úloha, je KS_R označena jako *obsazená*; jinak je *volná*.

Inverze priorit

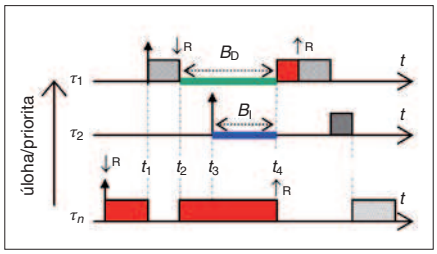
Další významný problém představuje pro systémy RT tzv. inverze priorit, ilustrovaná na *obr. 5* na příkladu úloh $\tau_1, \ldots, \tau_4$ s P_1 (nejvyšší) > … > P_4 (nejnižší), přičemž τ_2, τ_4 sdílejí prostředek R. Zaměřme se na dobu odezvy úlohy τ_2 – ta je delší o dobu potřebnou k provedení úlohy τ_1 s prioritou $P_1 > P_2$ (doba mezi t_3 a t_4) a dobu přímého blokování úlohy τ_2 úlohou τ_4 s prioritou $P_4 < P_2$ (doba mezi t_1 a t_5). Avšak doba přístupu τ_2 k R (tj. přítomnosti τ_2 v KS_R) je také prodloužena o dobu běhu úlohy τ_3, která ani nesdílí prostředek s τ_2, ani pro ni neplatí $P_3 > P_2$. V t_2 se totiž τ_2 stává nepřipravenou k běhu z důvodu nedostupnosti R a τ_3 je v t_2 jediná připravená úloha s prioritou větší než P_4. Úloha τ_3 tedy začíná běžet, čímž však odsouvá čas uvolnění R úlohou τ_4 až na t_4. Inverze priorit tedy nastává v intervalu mezi t_2, t_3 a t_4, t_5. Doba blokování prostředku R úlohou τ_4 – a tedy i doba odezvy úlohy τ_2 – je tudíž „zbytečně“ prodloužena o $(t_3 - t_2) + (t_5 - t_4)$ časových jednotek.

Uváznutí

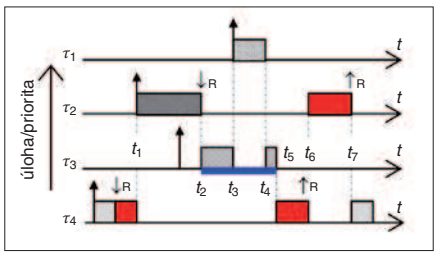
Jako poslední z problémů souvisejících se sdílením prostředků zmiňme uváznutí (*deadlock*). Jelikož, na rozdíl od předchozích problémů, může uváznutí vést k zastavení činnosti systému jako celku, je nutné mu předcházet zvlášť pečlivě. Uváznutí bude ilustrováno pomocí úloh τ_1, τ_2 sdílejících prostředky R_1, R_2 a přistupující k nim způsobem podle *obr. 6*. V čase t_1 vstoupí τ_2 do KS_{R1} (obsazenost KS_{R1} je zvýrazněna žlutou barvou). V t_2 přichází τ_1, přerušuje provádění τ_2; v t_3 vstupuje do KS_{R2} (její obsazenost je zvýrazněna červenou barvou) a v t_4 požaduje vstup do KS_{R1}, která je však stále obsazena úlohou τ_2 – ta ji pro přerušení úlohou τ_1 nestihla uvolnit. Úloha τ_1 se tedy stane nepřipravenou k běhu, procesor je jí odebrán a předán úloze τ_2. V t_5 požaduje τ_2 vstup do KS_{R2}, která je však obsazena úlohou τ_1. Je zřejmé, že každá z úloh τ_1, τ_2 by mohla pokračovat v běhu za předpokladu, že by druhá z nich uvolnila jí obsazenou KS. Nicméně to možné není, a dojde tedy k uváznutí (v *obr. 6* vyobrazeno modrou barvou). Obecně může uváznutí nastat i na množině n úloh $(\tau_1, \ldots, \tau_n)$, a to za předpokladu, že jsou současně splněny následující podmínky: pro $i = 1, \ldots, n$ platí: τ_i je v KS_{Ri} a čeká na uvolnění KS_{Rj} obsazené úlohou τ_j, kde $j = (i + 1) \bmod (n + 1)$, tj. τ_1 na τ_2, …, τ_n na τ_1 – čekání je cyklické, a k prostředkům se přistupuje výhradně v rámci příslušných KS (tj. zdroj R může uvolnit pouze úloha, která je v KS_R).

Protokoly přístupu k prostředkům

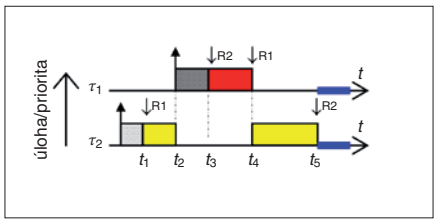
Zde bude představen přehled typických způsobů řešení již zmíněných problémů prostředky RTOS označovanými jako *protokoly přístupu k prostředkům*. Stručně budou zmíněny principy a vlastnosti protokolů NPP, PIP



Obr. 4. Ilustrace k blokování



Obr. 5. Ilustrace k inverzi priorit



Obr. 6. Ilustrace k uváznutí

a PCP. U každého z nich bude pozornost soustředěna na popis následujících skutečností: princip činnosti, požadavku kladeného na realizaci v RTOS, dopad na dobu blokování a schopnost zamezit již popsaným problémům.

Protokol NPP (nepreemptivní protokol, *Non Preemptive Protocol*) zamezuje preempci během provádění KS. Realizace NPP prostředky RTOS je jednoduchá – úloze τ vstupující do KS je, až do dokončení KS, zvýšena priorita na nejvyšší možnou. Výhoda tohoto přístupu je zřejmá: obsazená KS je vždy dokončena bez přerušení, tj. atomicky. Protokol NPP zamezuje inverzi priorit i uváznutí a ohraničuje shora dobu blokování. Jeho velkým nedostatkem je, že během provádění KS mohou být blokovány i úlohy, které mají prioritu významnější než P_τ (mimo KS by jistě vedly k přerušení provádění τ) a k žádnému prostředku nepřistupují.

Protokol PIP (protokol dědění priorit, *Priority Inheritance Protocol*) se snaží odstranit již uvedený nedostatek a omezit dopad protokolu pouze na úlohy přistupující k prostředkům. Princip PIP je následující: chce-li v čase t do KS obsazené úlohou τ_1 s prioritou P_1 vstoupit úloha τ_2 s $P_2 > P_1$, je P_1 změněna na P_2, a to od času t do času uvolnění KS. Předností protokolu PIP je, že odstraňuje nedokonalost protokolu NPP a ohraničuje shora dobu inverze priorit. Jeho nedostatkem je, že nezamezuje řetězenému blokování (tj. úloha s velkou prioritou přistupující k n prostředkům může být blokována n-krát) ani uváznutí.

Z hlavních nedostatků protokolu PIP vyplynuly snahy rozšířit PIP o schopnost zamezit uváznutí a řetězenému blokování. Princip tohoto rozšíření, známého pod označením PCP (protokol stropování/mezních priorit, *Priority Ceiling Protocol*), je následující: každému prostředku R, potenciálně používanému úlohami $\tau_1, \ldots, \tau_m$, je přiřazena tzv. *stropová*, popř. *mezní priorita* (P_R) rovná největší z hodnot $P_1, \ldots, P_m$. Při snaze o přístup do obsazené KS dědí úloha v KS prioritu podle PIP. Řetězenému blokování a uváznutí zamezuje následující pravidlo: úloha τ může vstoupit do KS_R pouze tehdy, je-li KS_R volná a $P_\tau > P_{max}$, kde $P_{max} = \max(P_{R1}, \ldots, P_{Rk})$, je největší ze stropových priorit zdrojů $R_1, \ldots, R_k$ momentálně používaných ostatními úlohami. Důsledky PCP pro praxi mohou být shrnuty do těchto bodů:

a) úloha může být zablokována i před přístupem k neobsazenému zdroji (prevence řetězeného blokování),
b) úloha s velkou prioritou (τ_1) může být blokována nejvýše jednou úlohou s nízkou prioritou (τ_2) přistupující k témuž R,
c) po uvolnění KS_R úlohou τ_2 již úloha τ_1 nemůže být při přístupu k žádným prostředkům blokována (avšak τ_1 může být stále blokována úlohou τ_3 s $P_3 > P_1$; taková úloha však jistě žádný prostředek s τ_1 nesdílí).

Oproti protokolu PIP zamezuje PCP řetězenému blokování i uváznutí. Základní vlastnosti protokolů NPP, PIP a PCP jsou uvedeny v *tab. 2*.

Jednoduchou modifikací PCP je možné získat protokol IPCP (okamžité PCP, *Immediate PCP*), jehož princip je následující: nízká priorita úlohy τ_2 nacházející se v KS není zvýšena až v čase blokování úlohy τ_1 s vysokou prioritou při jejím přístupu do KS, ale už v době přístupu τ_2 do KS. V porovnání s PCP je realizace IPCP jednodušší, vede k menšímu počtu přepnutí kontextu, avšak z pohledu jedné úlohy je zapotřebí větší počet změn priority.

Shrnutí

Závislosti představené v tomto článku lze jednak modelovat ve volně dostupných nástrojích – např. TimesTool [8], Cheddar

[2] – či zkoumat v souvislosti s realizacemi konkrétních RTOS. I ty nejjednodušší z existujících RTOS totiž zpravidla obsahují alespoň některé ze zmíněných protokolů ve svých jádrech – např. v rámci prostředku *mutex*, zajišťujícího vzájemně výlučný přístup v jádrech FreeRTOS, či µC/OS-II je zaveden protokol PIP. Mimoto lze rozhraní protokolů PIP nebo PCP také nalézt např. v rámci rozšířené normy POSIX 1003.1c a využít dříve popsané mechanismy ke zkvalitnění návrhu nejen úloh RT, ale i úloh řízených operačními systémy kompatibilními s touto normou.

Další díl seriálu bude zaměřen na přehled mechanismů společného plánování periodických a neperiodických úloh, přičemž cílem bude probrat a představit různé přístupy k dosažení co nejlepšího kompromisu mezi procentem obsloužených neperiodických podnětů a včasným dokončením periodických úloh.

Poděkování

Práce byla podpořena Evropským fondem regionálního rozvoje (ERDF) v rámci projektu Centra excelence IT4Innovations (CZ.1.05/1.1.00/02.0070), projektem Výzkum informačních technologií z hlediska bezpečnosti (CEZ MSM 0021630528) a grantem BUT FIT-S-11-1.

Literatura:

[1] BUTAZZO, G. C.: *Hard Real-Time Computing Systems, Predictable Scheduling Algorithms And Applications*. Kluwer, 1997, pp. 181–221, ISBN 0-7923-9994-3.

[2] *The Cheddar Project: A Free Real Time Scheduling Analyzer* [on-line]. Dostupné z http://beru.univ-brest.fr/~singhoff/cheddar/.

[3] COTTET, F. – DELACROIX, J. – KAISER, C. – MAMMERI, Z.: *Scheduling in Real-Time Systems*. John Wiley & Sons, 2002, ISBN 0-470-84766-2.

[4] STRNADEL, J.: *Návrh časově kritických systémů I: specifikace a verifikace*. Automa, 2010, roč. 16, č. 10, s. 42–44, ISSN 1210-9592.

[5] STRNADEL, J.: *Návrh časově kritických systémů II: úlohy reálného času*. Automa, 2010, roč. 16, č. 12, s. 18–19, ISSN 1210-9592.

[6] STRNADEL, J.: *Návrh časově kritických systémů III: priorita úloh*. Automa, 2011, roč. 17, č. 2, s. 50–52, ISSN 1210-9592.

[7] STRNADEL, J.: *Návrh časově kritických systémů IV: realizace prostředky RTOS*. Automa, 2011, roč. 17, č. 4, s. 58–60, ISSN 1210-9592.

[8] *The Times Tool* [on-line]. Dostupné z <www.timestool.com>.

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

EMO Hannover 2013 ve znamení výrobní inteligence

Strojírenské podniky se musí v současnosti vypořádat s mnoha obtížnými úkoly: často se mění objem výroby, výrobky jsou dodávány ve velkém množství variant, aby přesně splňovaly požadavky zákazníka, na výrobce je vyvíjen velký tlak na snižování ceny a složitější výrobky vznikají v různých závodech rozmístěných nezřídka velmi daleko od sebe. Tyto úkoly pomáhá vyřešit tzv. výrobní inteligence.

Co obsahuje pojem výrobní inteligence? Především je to automatizace výroby, a to její hardware i software. Ale obsah výrobní inteligence je širší než jen automatizace výroby. Zahrnuje také automatizovanou, počítačem podporovanou konstrukci a přípravu výroby (CAD/CAM, PLM), sběr dat a dokladování průběhu výrobních procesů (SCADA), podporu údržby a servisu (*asset management*) a vazby na ekonomické systémy řízení podniku a dodavatelských řetězců (MES). Do výrobní inteligence patří i celá komunikační a informační infrastruktura podniku.

Obr. 1. Veletrh EMO se bude konat v Hannoveru od 16. do 21. září 2013; uzávěrka přihlášek je v prosinci 2012

S tím vším se budou moci seznámit návštěvníci veletrhu obráběcích a výrobních strojů a linek EMO 2013, který se bude konat v Hannoveru od 16. do 21. září 2013 a jehož zdůrazněným tématem bude právě výrobní inteligence v celé její šíři. Proč toto téma na veletrhu, kde vystavují převážně dodavatelé výrobních strojů? Protože tito dodavatelé musí myslet nejen na výrobní funkce stroje, ale také na to, aby obsluha strojů a výrobní technici mohli spolehlivě a účinně plnit rostoucí požadavky na efektivní využívání materiálů a energie, zajištění kvality produkce, dokumentování výrobních procesů, sledování činnosti strojů a jejich prediktivní údržbu. Které nástroje k tomu mohou výrobnímu personálu pomáhat? Multimediální prvky v ovládání výrobních systémů, diagnostické systémy využívající webové služby, teleservis, technická podpora na dálku atd. Začlenění inteligentních a multimediálních ovládacích a asistenčních funkcí do řídicích systémů výrobních strojů a zařízení je zákazníky vyžadováno stále častěji a stává se významnou výhodou na trhu.

Veletrh EMO se koná až za rok, ale jeho přípravy, koncepční i organizační, jsou již v plném proudu. V současné době odchází do více než šedesáti zemí světa přihlášková dokumentace; uzávěrka přihlášek je v prosinci 2012. Ti, kteří na EMO dosud nevystavovali a přihláškovou dokumentaci nedostanou automaticky, si ji mohou vyžádat prostřednictvím webové stránky www.emo-hannover.de.

(Bk)