Paralelní programování

cvičení

Jan Outrata

únor–duben 2011

Cvičení 1

Jazyk C

POSIX Threads (UNIX)

- hlavičkový soubor pthread.h, knihovna libpthread
- funkce pthread_create, pthread_join, pthread_cancel aj.

Win32 API (MS Windows)

- hlavičkový soubor windows.h
- funkce CreateThread, WaitForSingleObject, GetExitCodeThread, TerminateThread, SuspendThread, ResumeThread aj.

Jazyk/Platforma C#/.NET

- jmenný prostor System
- třída Thread, metody run, start, join, yield aj.
- globální proměnné . . . statické členy ve třídě

Úkol (1 bod): Spusťte proceduru v **novém vlákně**, které předáte 2 textové řetězce a ona vrátí jejich spojení. Hlavní vlákno programu počká na dokončení nového vlákna a vypíše jeho návratovou hodnotu.

Cvičení 2

Úkol (1 bod): Sestavte scénáře pro program Počítadlo z 2. přednášky, ve kterých je výsledná hodnota $n = 10$ a $n = 2$.

Úkol (2 body): Naprogramujte program z předchozího úkolu a pokuste se demonstrovat sestavené scénáře. V rámci řešení vytvořte proceduru/metodu/makro **concurrent**, která/které bude mít za parametry libovolný počet (pouze) procedur, které spustí v samostatných vláknech a počká na jejich dokončení. Pro demonstraci scénářů může být potřeba vynucení možnosti výběru jiného vlákna (např. pauza/pozastavení vlákna na zadaný čas).

Cvičení 2

Úkol (3 body):

N **procesů, identifikátor procesu v proměnné** i	
	int $C[N] \leftarrow N$ různých hodnot
	int $D[N]$
	int $x, count$
1:	$x \leftarrow C[i]$
2:	$count \leftarrow$ počet prvků C menších než x
3:	$D[count + 1] \leftarrow x$

Co program dělá? Co by se stalo, kdyby pole C bylo inicializováno na hodnoty, které nejsou navzájem různé? Opravte program pro tento případ.

Cvičení 3

Zadání úkolů na cvičení je na slajdech z přednášky.

Úkol (2 body): Cvičení 1 ze slajdů.
Úkol (2 body): Cvičení 2 ze slajdů.
Úkol (1 bod): Cvičení 3 ze slajdů.

Další tři příklady ze slajdů jsou bonusové.

Cvičení 4

Úkol (2 body): Zjistěte, jakým způsobem jsou v POSIX Threads (UNIX)/Win 32 API (MS Windows) nebo na platformě .NET poskytovány složené atomické akce **test-and-set** a **exchange** a pomocí jedné z nich naprogramujte řešení problému kritické sekce.

Úkol (3 body): Dokažte **korektnost** řešení problému kritické sekce se složenou atomickou akcí test-and-set nebo exchange pomocí stavového diagramu.

Úkol (4 body): Vyřešte problém kritické sekce s použitím složené atomické akce **fetch-and-add** nebo **compare-and-swap**.

Cvičení 5

Zadání úkolů na cvičení je na slajdech z přednášky.

Úkol (1 bod): Cvičení 1 ze slajdů.
Úkol (3 body): Cvičení 2 ze slajdů. Pokud nebudou při převodu zamčeny současně oba účty, 1 bod.
Úkol (1 bod): Cvičení 4 ze slajdů.

Ostatní příklady ze slajdů jsou bonusové.

Cvičení 6

Úkol (2 body): Implementujte algoritmus na paralelní prefixy pro libovolno u zadanou operaci.
Úkol (3 body): V jednorozměrném poli `A` (indexovaným od 0) jsou zakódovány seznamy tak, že každá položka obsahuje index následujícího prvku seznamu. Poslední prvek seznamu obsahuje číslo -1. Napište funkci, která paralelně do prvků pole `B` uloží délky příslušných seznamů z pole `A`.
Úkol (2 body): Implementujte paralelní algoritmus, který zkomprimuje hustý vektor do řídkého vektoru.

Bonusové příklady
Úkol: Implementujte paralelní algoritmus pro skalární součin dvou hustých vektorů, resp. řídkého a hustého vektoru.
Úkol: Řídká matice je reprezentována polem řádků, které jsou reprezentovány jako řídké vektory. Implementujte paralelní algoritmus pro násobení řídké matice hustým vektorem.

Cvičení 7

Úkol (1 bod): Zjistěte, jakým způsobem jsou v POSIX Threads (UNIX)/Win 32 API (MS Windows) nebo na platformě .NET poskytovány (obecný) **semafor** a **mutex** (binární semafor).

Úkol (2 body): Naprogramujte řešení problému **producenta a konzumenta** pro lib. počet procesů producenta i konzumenta. Je nutné upravit řešení pro jedny procesy a pokud ano, jak?

Úkol (3 body): Vyřešte a naprogramujte řešení problému **bariéry** pro lib. počet procesů používající jediný (obecný) semafor.

Cvičení 7

Úkol (4 body, bonus):

Kritická sekce pro max. k z N procesů v krit. sekci

mutex $M \leftarrow 1, delay \leftarrow 0$
int $count \leftarrow k$

```
     int m
     loop forever
 1:      nekritická sekce
 2:      wait(M)
 3:      count ← count − 1
 4:      m ← count
 5:      signal(M)
 6:      if m ≤ −1
 7:         wait(delay)
 8:      kritická sekce
 9:      wait(M)
10:      count ← count + 1
11:      if count ≤ 0
12:         signal(delay)
13:      signal(M)
```

Sestavte scénář pro $N = 4, k = 2$, ve kterém bude $delay = 2$ v rozporu s definicí mutextu (jako binárního semaforu). Ukažte také pro $N = 3, k = 2$.

Cvičení 8

Úkol (3 body): Ověřte korektnost následující **simulace obecného semaforu**:

- inicializace na hodnotu $k \geq 0$: $M \leftarrow 1, gate \leftarrow 0, count \leftarrow k$

	wait
1:	wait(M)
2:	$count \leftarrow count - 1$
3:	if $count < 0$
4:	signal(M)
5:	wait(gate)
6:	else signal(M)

	signal
1:	wait(M)
2:	$count \leftarrow count + 1$
3:	if $count \leq 0$
4:	signal(gate)
5:	signal(M)

Cvičení 8

Úkol (5 bodů): **Problém stabilních** (či spíše ne nestabilních) **manželství**: N mužů a N žen, každý muž hodnotí ženy a každá žena hodnotí muže podle preferencí, párování jako seznam N párů (m, z), kde m je muž a z je žena, je *stabilní*, když pro lib. dva páry (m_1, z_1) a (m_2, z_2) platí, že m_1 preferuje z_1 před z_2 nebo z_2 preferuje m_2 před m_1. Najděte stabilní párování.

Např. pro preference klesající zleva doprava:

muž	ženy
1	2 4 1 3
2	3 1 4 2
3	2 3 1 4
4	4 1 3 2

žena	muži
1	2 1 4 3
2	4 3 1 2
3	1 4 3 2
4	2 1 4 3

existují pouze dvě možná stabilní párování:
$(1,4),(2,3),(3,2),(4,1)$ a
$(1,4),(2,1),(3,2),(4,3)$.

Vyřešte a naprogramujte řešení problému se samostatným procesem pro každého muže a každou ženu. Při řešení vyjděte ze sekvenčního algoritmu Gale-Shapley na Wikipedii (pod heslem „stable marriage problem“).

Cvičení 11

Úkol (1 bod): Zjistěte, zda a jakým způsobem jsou v POSIX Threads (UNIX)/Win 32 API (MS Windows) nebo na platformě .NET poskytovány **monitor**, **podmíněná proměnná** a **chráněný objekt**.

Úkol (2 body): Naprogramujte řešení některého **problému z přednášky 11** pomocí monitoru (s podmíněnými proměnnými) nebo chráněného objektu.

Úkol (3 body): Vytvořte **simulaci** monitoru pomocí semaforů, simulaci podmíněné proměnné pomocí (pasivních) semaforů a simulaci chráněného objektu pomocí monitoru (s podmíněnými proměnnými).

Úkol (3 body, bonus každý problém): Naprogramujte řešení některého **problému z přednášky 9, kromě večeřících filozofů**, pomocí monitoru (s podmíněnými proměnnými) nebo chráněného objektu.

Cvičení 12

Úkol (3 body): Naprogramujte **paralelní násobení matic**, tj. každý prvek výsledné matice je vypočítán v samostatném procesu, s použitím API OpenMP (v C) nebo Parallel Extensions/TPL (v C#/.NET).

Úkol (5 bodů, bonus): Předchozí úkol s použitím frameworku OpenCL. (Instalaci frameworku může být nutné provést do virtuálního systému, viz `http://site.inf.upol.cz/virtual_network.html`.)