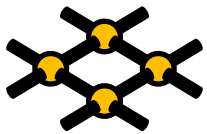
Vláknové programování
část I

Lukáš Hejmánek, Petr Holub
{xhejtman,hopet}@ics.muni.cz



Laboratoř pokročilých síťových technologií

PV192
2015–04–07

Vláknové programování v C/C++

1. Procesy, vlákna, přepínání kontextu, knihovna pthreads, vznik a ukončení vláken, základy ladění aplikací
2. Základy synchronizace: zámky, semafory, podmíněné proměnné
3. Pokročilé synchronizace: bariéry, rw zámky, pojmenované semafory, futexy
4. Afinita,Atributy vláken, režimy startu vlákna, priority, ukončování vláken, thread-specific data
5. OpenMP
6. Práce s pamětí
7. GUI, OpenGL, Futures a TPE v C++

Přehled přednášky

Procesy a vlákna
- Procesy
- Vlákna
- Pthreads
- OpenMP
- C++11

Procesy

- Proces
  - Instance programu, která je sekvenčně prováděna.
  - Je to entita pro alokace zdrojů (procesor, paměť, atd)
  - Procesy tvoří stromovou hierarchii–vztah rodič potomek

> Sex is not really common among processes—each process has just one parent.

Procesy

- Typy procesů
  - Levný proces (Light Weight Process–LWP)
    - Levné procesy mezi sebou sdílí adresní prostor
    - Minimum privátních zdrojů
  - Drahý proces (Heavy Weight Process–HWP)
    - Drahé procesy jsou mezi sebou zcela izolované
    - Prakticky všechny zdroje jsou privátní

Procesy

- Popisovač procesu
  - Obsahuje informace o
    - Signálech
    - Přidělené paměti
    - Otevřených souborech
    - Aktuálním adresáři
    - HW kontext (obsah registrů, zásobník, . . .) – TSS
    - Terminálu
    - Prioritě
    - Stav
    - . . .

Procesy

- Vytvoření procesu
  - Proces vzniká rozštěpením rodiče
  - Po startu je potomek stejný jako rodič
    - Stejný obsah paměti (Copy on Write)
    - Vykonává stejný kód

Procesy

- Běh procesu
  - Vykonávání kódu programu – charakterizován kontextem
  - Dva režimy běhu
    - User space–kód samotného programu
    - Kernel space–kód jádra
  - Stav procesu
    - Running
    - Interruptible
    - Uninterruptible
      (Nezpracovává signály)
    - Stopped
    - Traced
  - Konkurence vs. paralelismus
    - Konkurence–vykonávání stejného nebo různého kódu více procesy, nemusí probíhat ve stejný čas
    - Paralelismus–konkurence probíhající ve stejný čas

Procesy

- Přepínání kontextu
  - Zásadní mechanismus multitaskingu
  - Mechanismus uložení a obnovení stavu CPU
  - Rozlišujeme přepnutí kontextu
    - Registrové (obsluha přerušení)
    - Vláknové (přepnutí na jiné vláko téhož procesu)
    - Procesové (přepnutí na jiný proces)

Procesy

- Kroky při přepínání kontextu
  - Uložení stavu CPU, obvykle do TSS
    - Všechny běžné registry, deskriptory segmentu, příznaky
    - Stav a registry FPU
  - Obnova adresního prostoru
  - Načtení nového stavu CPU

Procesy

- Softwarové vs. hardwarové přepínání kontextu
  - Kontext lze uložit a obnovit v softwaru (kopírování stavu)
  - Některé procesory podporují přepnutí kontextu v HW (architektura x86 od Intel 80386 a dál)
  - Linux od verze jádra 2.4 používá softwarové přepnutí kontextu
  - Softwarové i hardwarové přepnutí kontextu je velmi drahá operace!

Procesy

- Komunikace mezi procesy
  - Soubory
  - Signály (signal(7))
  - Sockets (socket(2))
  - Fronty zpráv (mq_overview(7))
  - Trubky (pipes), pojmenované vs. nepojmenované (pipe(2))
  - Semafory (sem_overview(7))
  - Sdílená paměť, paměťově mapované soubory (shm_overview(7), mmap(2))
  - Message passing (MPI knihovny)

Vlákna

- Proč vlákna?
  - Paralelismus
  - Výkon
  - Odezva
  - Komunikace

Vlákna

- Vlákno
  - Podmnožina procesu
  - Vlákno nemá vlastní adresní prostor
  - Typicky sdílí stav ostatními vlákny daného procesu
  - Shared-memory model:
    - Komunikace mezi vlákny je možná stejně jako u procesů a navíc i přes jejich sdílenou paměť
    - Oproti procesu jsou běžně sdílené globální a statické proměnné

Vlákna

- Vlákna stejného procesu sdílí
  - Kód programu
  - „Většinu“ dat
    - Novější koncepce vláken podporuje nesdílenou paměť–thread local storage (TLS)
  - Otevřené soubory (file descriptors)
  - Signály a obsluhu signálů
  - Současný pracovní adresář
  - Identifikaci uživatele a skupiny
  - Process ID (PID)
  - Pojmenované semafory, fronty zpráv a další nástroje IPC

Vlákna

- Každé vlákno má unikátní
  - Thread ID (identifikace vlákna)
  - Obsah registrů procesoru, ukazatel vrcholu zásobníku
  - Zásobník pro lokální proměnné a návratové adresy
  - Masku signálů
  - Prioritu
  - Hodnotu proměnné **`errno`** (dle POSIX.1c)

Vlákna

- Implementace vláken v operačním systému je různá
  - proces a vlákno není rozlišeno
    (vlákno je tedy procesem – Linux bez NPTL)
  - proces a vlákno jsou rozlišeny
    (vlákno se liší od procesu – Windows, Linux s NPTL)
  - vlákna v uživatelském prostoru
    (vlákna si řídí sám proces – Java Green Threads (obsolete), Erlang)

Vlákna

- Mapování vláken na plánovací entity v jádře
  - mapování vláken 1:1
    - současné produkční implementace
    - Linux, Windows, FreeBSD s libthr
  - mapování vláken N:M
    - \+ teoreticky nejefektivnější
    - − příliš složité, problémy s invertováním priorit, atd.
    - FreeBSD s Kernel Scheduler Entitites, experimenty i v Linuxu
  - mapování vláken N:1
    - zastaralý přístup, user-space threading
    - FreeBSD s libc_r

Vlákna

- Některé systémy mají „levné“ přepínání vláken a „drahé“ přepínání procesů (Windows NT, OS/2).
- Některé systémy příliš nerozlišují v přepnutí vlákna nebo procesu.
- Proč uvažujeme vlákna místo procesů?
  - Rychlejší přepnutí běžících vláken než běžících procesů.
  - Snadné sdílení paměti a dalších zdrojů mezi běžícími vlákny (někdy ovšem nevýhoda).

Knihovna Pthreads

- POSIX Threads (Pthreads) je POSIX standard pro vlákna
- Vlákna v systémech na bázi jádra Linux
  - Linux threads – neúplná implementace POSIX Threads
    - Vlákno bylo obsluhováno stejně jako proces, mělo i vlastní PID (process ID).
    - Není nutná speciální podpora jádra, problémy s výkonem, pokud se vlákno samo nevzdá procesoru (yield()).
  - Nahrazena NPTL – Native POSIX threads library
    - Výrazně vyšší výkonnost
    - Vlákno je samo o sobě jednotkou plánování, tj. procesový plánovač plánuje i vlákna obvykle úplně stejně.
    - NPTL potřebuje speciální podporu jádra pro synchronizaci.
- Pthreads knihovna má implementaci pro řadu systémů: Linux, *BSD, Windows, MacOS, ...

Kompilace

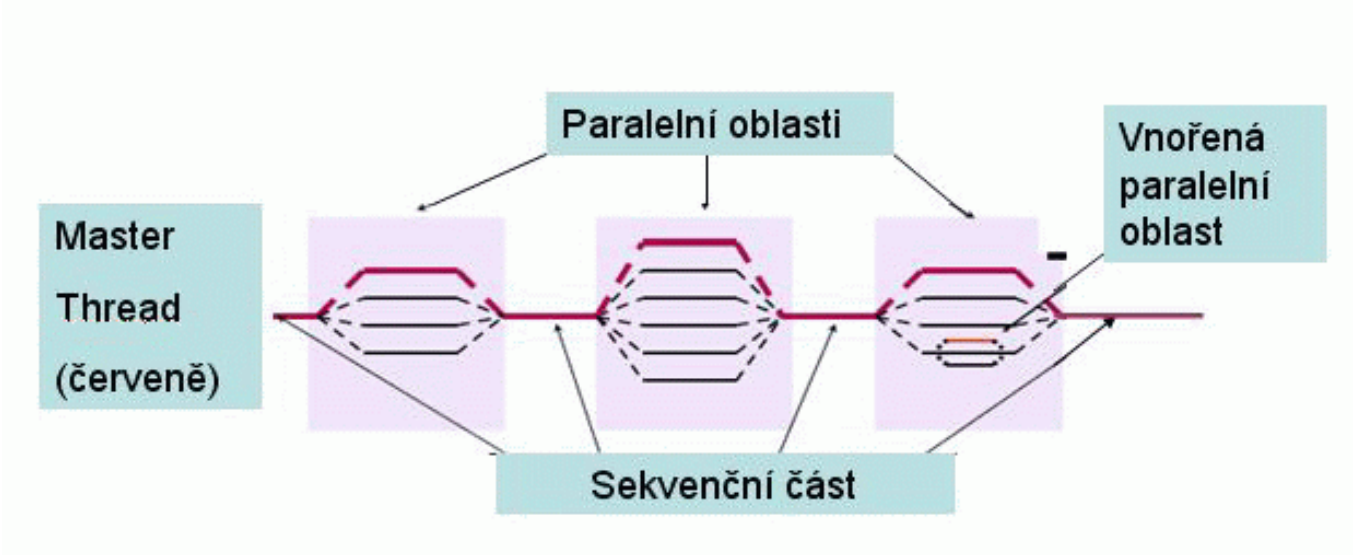
- Dvě možnosti kompilace:
  - `gcc -o foo foo.c -lpthread -D__REENTRANT`
  - `gcc -o foo foo.c -pthread -D__REENTRANT`
- Nezapomínáme na to, že záleží na pořadí knihoven a objektových souborů na příkazové řádce.

Open MP

- Standard pro programování se sdílenou pamětí
- Podpora v programovacích jazycích Fortran (od r. 1997), C, C++ (od r.1998)
- Současná verze 3.0 je z roku 2008 pro Fotran i C/C++
- Podporováno řadou překladačů vč. gcc a Intel cc
- Podpora paralelního programování pomocí
  - Soustavy direktiv pro překladač
  - Knihovních procedur
  - Proměnných prostředí

Programovací model OpenMP

- Explicitní paralelismus
- Fork/join model



- Vnořený (nested) paralelismus není vždy dostupný funkcí)

Překlad

- **`gcc -g -o foo foo.c -fopenmp -D_REENTRANT`**
- Aplikace je slinkována s knihovnami **`libgomp`** a **`libpthread`**.

Přehled syntaxe

- Základní formát
  `#pragma omp jméno-příkazu [klauzule] nový_řádek`
- Všechny příkazy končí novým řádkem
- Používá konstrukci pragma (pragma = věc)
- Rozlišuje malá/velká písmena
- Příkazy mají stejná pravidla jako C/C++
- Delší příkazy lze napsat na více řádků pomocí escape znaku \

C++11

- Novější standard C++ publikovaný v září 2011
- Jako první verze obsahuje podporu vláken
- Následnící C++14 a C++17 (C++14 vydán v srpnu 2014)

Překlad C++11

- **`g++ -o c11test c11test.C -std=c++11 -pthread`**
- Neuvedení **`-pthread`** způsobí runtime chybu

```
terminate called after throwing an instance of 'std::system_error'
  what():  Enable multithreading to use std::thread: Operation not
 permitted
Aborted (core dumped)
```