Operační systémy 1

10. Cvičení

Radek Janoštík

KATEDRA
INFORMATIKY
UNIVERZITA PALACKÉHO V OLOMOUCI

Úkoly z minula – jak to šlo?

- Kolik se vám toho podařilo udělat?
  1. Vypsání `pid`, zjištění `pid`, ukončení programu
  2. `fork` a `exec`
  3. `kill` a čekání
- Dalo něco zabrat?
- Nějaké dotazy?
- Prezenčka (po probrání látky)

Vlákna ve Windows

- *Vlákno* = základní jednotka vykonávající instrukce programu
- *Proces* je tvořen jedním nebo více vlákny
  - *Primární vlákno* – při spuštění procesu, funkce `main`
  - Vlákno má vlastní kontext (registry, zásobník)
  - Sdílení prostředků (paměť, systémové prostředky, . . .)
- Běh vláken plánovat OS $\Rightarrow$ „nepředvídatelné“ libovolné prokládání
- Potřeba řízení přístupu ke sdíleným prostředkům – synchronizace

Vytvoření vlákna

- Vytvoření vlákna zajistí funkce `CreateThread`
  - Kód k vykonání předáváme odkazem na funkci
  - Data (argumenty) předáváme také odkazem
  - Id vytvořeného vlákna zapíše na místo, kam ukazuje poslední parametr
  - Vrací `HANDLE` (pro budoucí manipulaci)
- Měli bychom zkontrolovat, zda vůbec došlo k vytvoření vlákna
  - `HANDLE` může být `NULL`
- Měli bychom počkat na dokončení práce vlákna
- A práci řádně ukončit
- Ukázka kódu

Manipulace s vlákny

- `GetCurrentThread` – zjištění pseudo-handle aktuálního vlákna
  - pseudo-handle má platnost pouze v aktuálním vlákně
  - pro „globální použití“ nutno převést na `HANDLE` pomocí `DuplicateHandle`
- `SuspendThread` – uspání vlákna
  - Může být voláno opakovaně – zvýšení počítadla
- `ResumeThread` – probuzení vlákna
  - Snížení počítadlo
  - Je-li počítadlo vynulováno, dojde k probuzení vlákna
- `ExitThread` – ukončí právě prováděné vlákno
- `TerminateThread` – ukončí jiné vlákno
  - `HANDLE` jako argument
  - K ukončení dojde „kdekoliv“, ukončované vlákno nemá možnost zareagovat
  - $\Rightarrow$ program může skončit v nekonzistentním stavu

Úkol (1/2)

1. Odstraňte z kódu volání `Sleep`, případně změňte jeho argumenty, a sledujte, jak se změní průběh programu.

2. Rozšiřte ukázkový program tak, aby vyzvedl a vypsal návratovou hodnotu spuštěného vlákna. Zamyslete se nad tím, kam volání funkce `GetExitCodeThread` umístit.

3. Rozšiřte ukázkový program, aby pracoval obecně s `N` vlákny, kde `N` je konstanta zadaná v kódu programu.

Úkol (2/2)

4. Ukázkový příklad upravte tak, že po provedení `(COUNT / 2)` cyklů se vlákno uspí a je probuzeno v momentě, kdy primární vlákno zpracuje celou smyčku.

5. Naprogramujte funkci `int parfib(int)`, která spočítá Fibonacci číslo rekurzivním způsobem s využitím právě dvou vláken. Dvě počáteční větve výpočtu spusťte v samostatných vláknech.

6. Naprogramujte funkci `int pmin(int *numbers, unsigned int count, unsigned int threads)`, která použije `threads` vláken k tomu, aby v poli `numbers`, které obsahuje `count` hodnot, našla nejmenší hodnotu.