KATEDRA
INFORMATIKY
UNIVERZITA PALACKÉHO V OLOMOUCI

Základní práce s vlákny v Linuxu

KMI/OS1 Operační systémy I

Mgr. Markéta Trnečková, Ph.D.
www.marketa-trneckova.cz

Základní práce s vlákny ve Windows

Opakování

Příklad

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

Příklad

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

Základní práce s vlákny ve Windows

Opakování

Příklad

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

- Jak jste řešili synchronizaci a předávání výsledků?

Základní práce s vlákny v Linuxu

Učební text ke cvičení:
`https://phoenix.inf.upol.cz/~krajcap/courses/2025LS/OS1/tutorial11.pdf`

Vlákna

- v unixových systémech je historicky základní jednotkou vykonávající program proces
- později byla vlákna do unixových systémů přidána (protože se osvědčila)
- práce s nimi je podobná jako ve Windows (viz předchozí cvičení)
- interně se v unixových systémech s vlákny podobně jako s procesy
- každé vlákno má svou sadu registrů, zásobník atd.
- paměťový prostor a systémové prostředky jsou sdíleny v rámci jednoho procesu

Vytvoření vlákna

- **vytvoření vlákna** pomocí funkce `pthread_create`

```
int pthread_create(
                    pthread_t *thread,
                    pthread_attr_t *attr,
                    void *(*start_routine)(void*),
                    void *arg)
```

- argumenty:
  - ukazatel na hodnotu typu `pthread_t` – obsahuje identifikátor vytvořeného vlákna a umožňuje s tímto vláknem pracovat
  - atributy, které má vytvořené vlákno mít (v případě, že použijeme `NULL`, použije se výchozí nastavení)
  - funkce, která představuje kód, který se má vláknem vykonávat
  - tato funkce vrací `void *`, který je vláknu předán při jeho vytvoření (4. argument)

Vytvoření vlákna

Příklad

```c
#include <stdio.h>
#include <pthread.h>
#include <time.h>

#define COUNT (20)

static void msleep(int ms){
    struct timespec t;
    t.tv_sec = ms / 1000;
    t.tv_nsec = (ms % 1000) * 1000000;
    nanosleep(&t, NULL);
}

static void *thread_func(void *arg) {
    int id = *((int *) arg);
    printf("Spusteno vlakno: %i\n", id);
    for(int i = 0; i < COUNT; i++) {
            printf("thr #%i: %i\n", id, i);
            msleep(5);
    }
    return (void *)42;
}

int main() {
    int id = 1;
    long result;
    pthread_t thread;
    if(pthread_create(&thread, NULL, thread_func, &id)) {
            fprintf(stderr, "Vytvoreni vlakna selhalo\n");
            return 1;
    }

    for(int i = 0; i < COUNT; i++) {
            printf("thr #main: %i\n", i);
            msleep(5);
    }

    pthread_join(thread, (void **) &result);
    printf("Result: %li\n", result);
    return 0;
}
```

Vytvoření vlákna

- `pthread_create` – argument `thread_func` funkce, kterou bude vlákno vykonávat
- pokud vytvoření vlákna selže, je vrácena nenulová hodnota
- s vláknem můžeme pracovat pomocí hodnoty, která je určena prvním argumentem
- pro čekání na dokončení běhu vlákna slouží funkce `pthread_join`, ta čeká na doběhnutí vlákna a slouží k vyzvednutí návratové hodnoty funkce vlákna
- kód dělá v podstatě to samé, co minulý ukázkový kód
- místo funkce `Sleep`, zde používáme `msleep`, která uspí vlákno na zadaný počet milisekund

Pozor! Tento způsob práce s vlákny stojí mimo standardní knihovnu jazyka C, ke potřeba při překladu použít přepínač `-pthread`, nebo připojit knihovnu `libpthread` a použít přepínač `-lpthread`.

Viz přiložený Makefile.

Ukončení vlákna a uvolnění prostředků

- všimněme si, že v kódu neuvolňujeme prostředky s vláknem spojené (chybí ekvivalent `CloseHandle`
- o uvolnění prostředků se stará `pthread_join`
- pokud bychom chtěli vlákno spustit, aniž by nás zajímala vrácená hodnota, nemáme kam napsat volání `pthread_join`
- v takovém případě vytvoříme vlákno s atributem `PTHREAD_CREATE_DETACHED`
- nebo tento atribut nastavit za běhu funkcí `pthread_detach`
- pokud to chceme nastavit u právě běžícího vlákna, můžeme použít `pthread_self`, která vrací identifikátor aktuálně běžícího vlákna

Příklady

Příklad

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

Příklad

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

Příklad

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

Příklad

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.