

Vláknové programování

část IV

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



Laboratoř pokročilých síťových technologií

PV192
2013-04-23

Přehled přednášky

Ukončování

TSD

Ukončování

Ukončování

- Dvě varianty ukončení:
 - Samotným vláknem
 - `pthread_exit()`.
 - Návrat z hlavní funkce vlákna.
 - Jiným vláknem
 - `pthread_kill()`
 - `pthread_cancel()`

pthread_cancel ()

- **pthread_cancel ()** pošle danému vláknou notifikaci, aby se ukončilo.
- Vlákna mohou mít nastaveny dva různé typy kancelace:
 - **PTHREAD_CANCEL_DEFERRED** – vlákno je ukončeno pouze v tzv. kancelačních bodech (default).
 - **PTHREAD_CANCEL_ASYNCHRONOUS** – vlákno je ukončeno okamžitě.
- Dále vlákna mohou kancellaci odmítnout **PTHREAD_CANCEL_DISABLE**, opětovně přijmout kancellaci jde pomocí **PTHREAD_CANCEL_ENABLE**.
- Typy kancellace nastavíme pomocí **pthread_setcanceltype ()**.
- Přijmout/odmítnout kancellaci lze pomocí **pthread_setcancelstate ()**.

Kancelační body

- Kancelační bod je volání funkce, ve které může být vlákno ukončeno, je-li typu **PTHREAD_CANCEL_DEFERRED**.
- Základní kancelační body jsou:
 - **pthread_testcancel()** – pouze zjistí, zda nebylo signalizováno *cancel*
 - **pthread_setcancelstate()** – pokud měníme stav z **PTHREAD_CANCEL_DISABLE** na **PTHREAD_CANCEL_ENABLE**, je volání kancelačním bodem.
- Další kancelační body:
 - **pthread_cond_wait()**, **pthread_cond_timedwait()**, **pthread_join()**, **sem_wait()** (pouze z knihovny pthreads, pokud je poskytnuta knihovnou libc, není to kancelační bod!).
 - Většina funkcí **libc** (zejména I/O funkce), je vhodné konzultovat dokumentaci.

Příklad na kancellaci

```
1 #include <pthread.h>
2
3 void *
4 foo(void *arg)
5 {
6     int old;
7     pthread_setcanceltype(PTHREAD_CANCEL_DEFERRED, &old);
8     while(1) {
9         pthread_testcancel();
10    }
11    return NULL;
12 }
13
14 int
15 main()
16 {
17     pthread_t t;
18
19     pthread_create(&t, NULL, foo, NULL);
20
21     pthread_cancel(t);
22
23     return 0;
24 }
```

Cleanup Push/pop

- Co dělat v případě, že vlákno, kterému posíláme cancel, zrovna drží nějaký zámek?
- **pthread_testcancel()** rovnou vlákno ukončí, nelze použít pro test a případně zámek odemknout.
- Push/pop
 - Vlákno má zásobník funkcí, které se mají provést v případě kancelace.
 - **pthread_cleanup_push()** přidá specifikovanou funkci na vrchol zásobníku.
 - **pthread_cleanup_pop()** odebere funkci z vrcholu zásobníku (lze říct, zda funkci rovnou provést).
 - Některé implementace pthreads hlídají párování push/pop pomocí maker a ke každému push v každé funkci musí být odpovídající pop!

Příklad na cleanup

```
1 #include <pthread.h>
2
3 pthread_mutex_t lock;
4
5 void *
6 foo(void *arg)
7 {
8     int old;
9     pthread_setcanceltype(PTHREAD_CANCEL_DEFERRED, &old);
10    pthread_cleanup_push(pthread_mutex_unlock, &lock);
11    pthread_mutex_lock(&lock);
12    while(1) {
13        pthread_testcancel();
14    }
15    pthread_cleanup_pop(1); /*execute unlock*/
16    return NULL;
17 }
18
19 int
20 main()
21 {
22     pthread_t t;
23
24     pthread_create(&t, NULL, foo, NULL);
25
26     pthread_cancel(t);
27     return 0;
28 }
```

Thread specific data

Thread-Specific Data

- Řada nástrojů pro paralelní běhy vláken umožňuje vytvořit privátní datovou oblast vlákna – TLS (Thread local storage).
- TLS je využito například knihovnou OpenGL (i když poněkud nešťastně) pro uchování kontextu.
- TLS je poskytnuto Javou, některými C/C++ variantami (GNU C, Intel C/C++, Visual C++, a další), C#, Python, Dephi.

TLS v Pthreads

- Princip použití:
 - Vytvoření klíče (s volitelným destruktorem).
 - Svázání klíče s nějakými daty.
 - Vyhledání dat podle klíče.
- Klíč je globální pro všechna vlákna daného procesu.
- Vazba dat na klíč je pro každé vlákno separátní.

TLS v Pthreads

- Datový typ klíče `pthread_key_t`.
- Vytvoření klíče `pthread_key_create()`.
 - Při vytváření klíče je možné specifikovat destruktory, který se zavolá v případě ukončení vlákna.
- Svázání dat a klíče `pthread_setspecific()`.
- Vyhledání dat dle klíče `pthread_getspecific()`.
- Zrušení klíče `pthread_key_delete()`.

Příklad na TLS

```
1 #include <pthread.h>
2 #include <stdio.h>
3 #include <stdlib.h>
4
5 pthread_key_t key;
6
7 void
8 msg(char *m)
9 {
10     char *buff = pthread_getspecific(key);
11     sprintf(buff, "%s\n", m);
12     printf(buff);
13 }
14
15 void *
16 runner(void *arg)
17 {
18     char *array;
19     int i;
20
21     array = malloc(20);
22     pthread_setspecific(key, array);
23     for(i = 0; i < 10; i++) {
24         msg(arg);
25     }
26     return NULL;
27 }
```

Příklad na TLS

```
27
28 int
29 main(void)
30 {
31     pthread_t t1, t2;
32
33     pthread_key_create(&key, free);
34
35     pthread_create(&t1, NULL, runner, "Hello");
36     pthread_create(&t2, NULL, runner, "Hello_world");
37
38     pthread_join(t1, NULL);
39     pthread_join(t2, NULL);
40
41     pthread_key_delete(key);
42     return 0;
43 }
```

Jednodušší použití

- Použití pomocí klíčů je trochu těžkopádné
- GCC nabízí (neportabilní) direktivu `__thread`
- Použití:
 - `__thread` proměnná
 - `__thread int x`
 - Má zde význam slovo `volatile`?

Příklad

```
1 #include <pthread.h>
2 #include <stdio.h>
3
4 __thread int x=0;
5
6
7 void *
8 worker(void *arg) {
9     for(;x<1000000;x++) {
10         asm volatile("":"m" (x));
11     }
12     printf("X_val:_%d,_addr_%p\n", x, &x);
13 }
14
15 int main()
16 {
17     pthread_t t[2];
18
19     pthread_create(&t[0], NULL, worker, NULL);
20     pthread_create(&t[1], NULL, worker, NULL);
21     pthread_join(t[0], NULL);
22     pthread_join(t[1], NULL);
23     printf("X_val:_%d,_addr_%p\n", x, &x);
24 }
```

- Příklad výstupu:
X val: 1000000, addr 0x7ff3966d470c
X val: 1000000, addr 0x7ff395ed370c
X val: 0, addr 0x7ff396e706fc