## Procesy, vlákna

**Vlákno**

- odlehčený proces
- kód vlákna, zásobník privátní
- ostatní sdíleno s dalšími vlákny téhož procesu

**Implementace vláken**

- one-to-one
  - implementace na úrovni jádra
  - každé vlákno je pro jádro samostatný proces
- many-to-one
  - implementace v uživatelském režimu prostřednictvím systémové knihovny
  - malá režie přepínání vláken
  - problém s během na multiprocesorovém systému (vlákna neběží zároveň)
- many-to-many
  - kombinace

Linux – první model – volání jádra clone(2) – nepoužíváno přímo, ale prostřednictvím knihoven obhospodařujících vlákna (libpthread)

## Tvorba vlákna a jeho ukončení

| | | |
|---|---|---|
| Vytvoření vlákna | - | pthread_create |
| Předčasné ukončení | - | pthread_exit |
| Čekání na ukončení vlákna | - | pthread_join |

## Parametry funkce pthread_create

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

př.: `pthread_create(&thread_id, NULL, &print_x, NULL);`

volání funkce se vrací okamžitě a původní vlákno pokračuje dále
nové vlákno začíná realizovat kód, obsažený ve funkci vlákna

1. Ukazatel na proměnnou typu pthread_t, do které se uloží identifikační číslo nově vytvořeného vlákna.
2. Ukazatel na objekt atribut vlákna (thread attribute). Tento objekt obsahuje detailní informace o tom, jak má vlákno komunikovat se zbývajícími komponentami programu. Pokud prostřednictvím tohoto parametru předáte NULL, budou nastaveny implicitní atributy.
3. Ukazatel na funkci vlákna. Jedná se o obyčejný ukazatel na funkci typu:
4. void *(*) (void *)
5. Argument vlákna pro přenos dat typu void *. Pomocí tohoto argumentu můžete předávat funkci jakákoliv data.

Příklad

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

/* Funkce vlákna - tiskne písmena x do stderr.
   Parametr není používán. Funkce se nevrací. */

void *print_x(void *unused)
{
  while(1)
    fputc('x', stderr);
  return(NULL);
}


int main()
{
  pthread_t thread_id;

  /* Vytvoří nové vlákno. Nové vlákno bude
     vykonávat funkci print_x(). */

  pthread_create(&thread_id, NULL, &print_x, NULL);

  /* Tiskne písmena o do stderr. */
  while(1)
    fputc('o', stderr);
  return(0);
```

Přeložení příkladu:

```
gcc -lpthread -o vlakna1 vlakna1.c
```

Další informace:

**Seriál na rootu o vláknech v C.**
http://www.root.cz/clanky/programovani-pod-linuxem-tema-vlakna/

**Tutoriál k POSIX threadům**
http://www.yolinux.com/TUTORIALS/LinuxTutorialPosixThreads.html

## Synchronizace

- Mutexy – blokuje přístup k proměnné jiným threadům
- Join – čekání na dokončení jiného vlákna
- Podmínkové proměnné

## Mutex

```
pthread_mutex_t mutex1 =
PTHREAD_MUTEX_INITIALIZER;

int counter=0;

void functionC()
{
   pthread_mutex_lock( &mutex1 );
   counter++
   pthread_mutex_unlock( &mutex1 );
}
```

Příklad mutex:

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>

void *functionC();
pthread_mutex_t mutex1 = PTHREAD_MUTEX_INITIALIZER;
int  counter = 0;


main()
{
    int rc1, rc2;
    pthread_t thread1, thread2;


   if( (rc1=pthread_create( &thread1, NULL, &functionC, NULL)) )
   {
      printf("Neslo vytvořit vlákno: %d\n", rc1);
   }

   if( (rc2=pthread_create( &thread2, NULL, &functionC, NULL)) )
   {
      printf("Nešlo vytvořit vlákno: %d\n", rc2);
   }


   /* Ćekáme na dokončení vláken.*/
   /* Jinak riskujeme exit a ukončení všech vláken */
   /* před jejich dokončením.                      */

    pthread_join( thread1, NULL);
    pthread_join( thread2, NULL);

   exit(0);
}

void *functionC()
{
   pthread_mutex_lock( &mutex1 );
   counter++;
   printf("Counter value: %d\n",counter);
   pthread_mutex_unlock( &mutex1 );
}
```

Podmínkové proměnné

- Proměnná typu pthread_cond_t
- Thread pozastaví vykonávání, dokud nějaká podmínka není true

Rozebrat a odzkoušet tento příklad (conditional variables)
http://www.yolinux.com/TUTORIALS/LinuxTutorialPosixThreads.html

## Thready v Javě

http://java.sun.com/docs/books/tutorial/essential/threads/timer.html

## implementace

Oddědit od třídy Thread, přepsat metodu run, spustit zavoláním start

```
public class SimpleThread extends Thread {
    public SimpleThread(String str) {
        super(str);
    }
    public void run() {
        for (int i = 0; i < 10; i++) {
            System.out.println(i + " " + getName());
            try {
                sleep((long)(Math.random() * 1000));
            } catch (InterruptedException e) {}
        }
        System.out.println("DONE! " + getName());
    }
}


public class TwoThreadsTest {
    public static void main (String[] args) {
        new SimpleThread("Jamaica").start();
        new SimpleThread("Fiji").start();
    }
}
```

Implementace Runnable interface
(nutné pokud naše třída musí dědit od jiné podtřídy)

http://cretesoft.com/archive/newsletter.do?issue=101

http://java.sun.com/docs/books/tutorial/uiswing/misc/threads.html

http://www.yolinux.com/TUTORIALS/LinuxTutorialPosixThreads.html