PB173 Linux

07 Vlákna

Roman Lacko xlacko1@fi.muni.cz

2022-11-04

Obsah

Vlákna

Vlákna

Vlákno je abstrakcia výpočtu (toku riadenia) v programe.

Proces je jedno alebo viac vláken, ktoré zdieľajú (niektoré) zdroje.

- Operácie nad procesom môžu ovplyvniť všetky vlákna.
- Všetky vlákna procesu sú rovnocenné.

Vlákna: Typy

Kernel Threads

- Tiež *Native Threads*
- Najmenšia jednotka plánovača
- Typicky *preemptívne plánovanie*

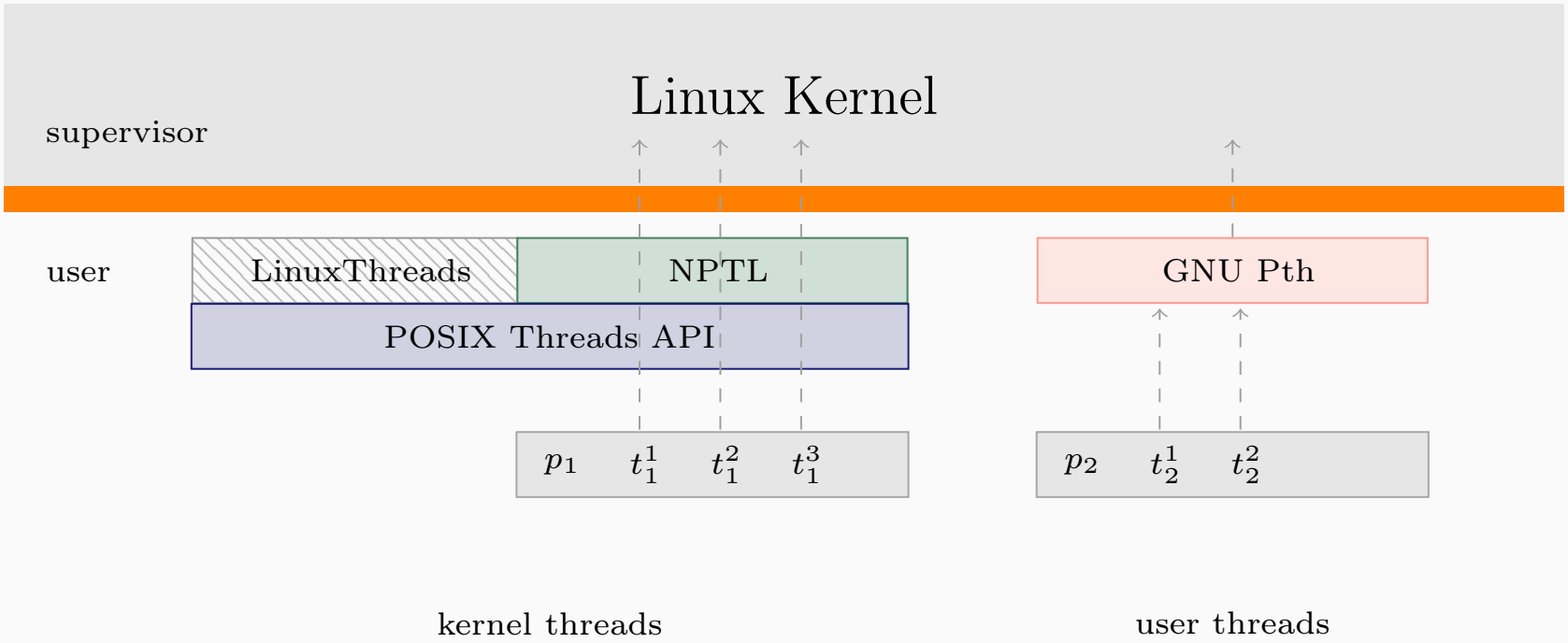
Pozor na nejednoznačnosť

Kernel Threads sú tiež vlákna, ktoré vykonávajú vedľajšie úlohy jadra v privilegovanom režime (napr. `kworker`).

User Threads

- Tiež *Green Threads*
- Môže a nemusí vyžadovať podporu jadra
- Typicky *kooperatívne plánovanie*

Vlákna


Linux Kernel
supervisor
user
LinuxThreads
NPTL
POSIX Threads API
GNU Pth
p_1 t_1^1 t_1^2 t_1^3
p_2 t_2^1 t_2^2
kernel threads
user threads

Vlákna: Terminológia

Proces sa skladá z *úloh*, ktoré postupne vykonávajú *akcie*.

Multitasking

Možnosť prepínať medzi práve spracovanými procesmi.

Konkurentný (súbežný) systém (s multitaskingom)

Umožňuje vykonávanie viacerých *úloh* procesu v jednom čase.

Paralelný systém (s multitaskingom)

Umožňuje vykonávanie viacerých *akcií* v jednom čase.

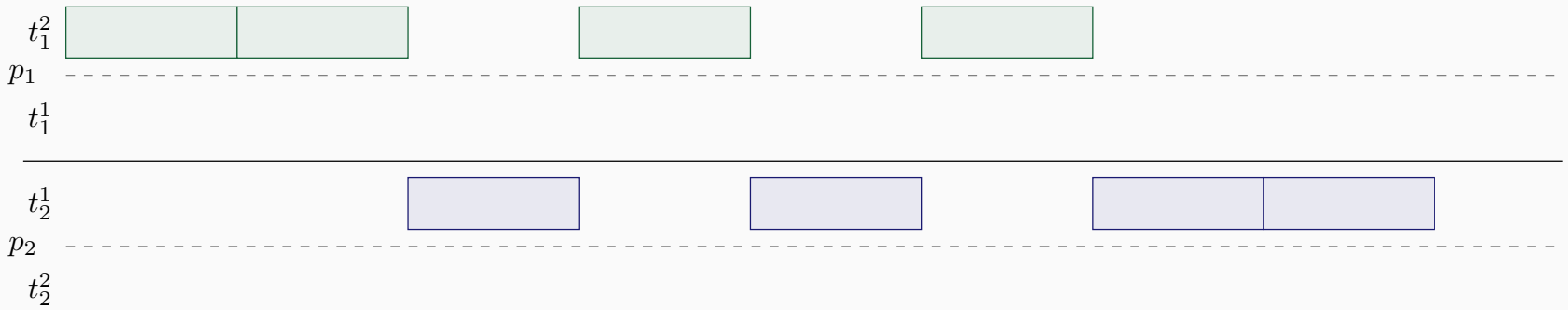
Formálne definície

Táto definícia je prispôsobená pre *multitasking OS*.

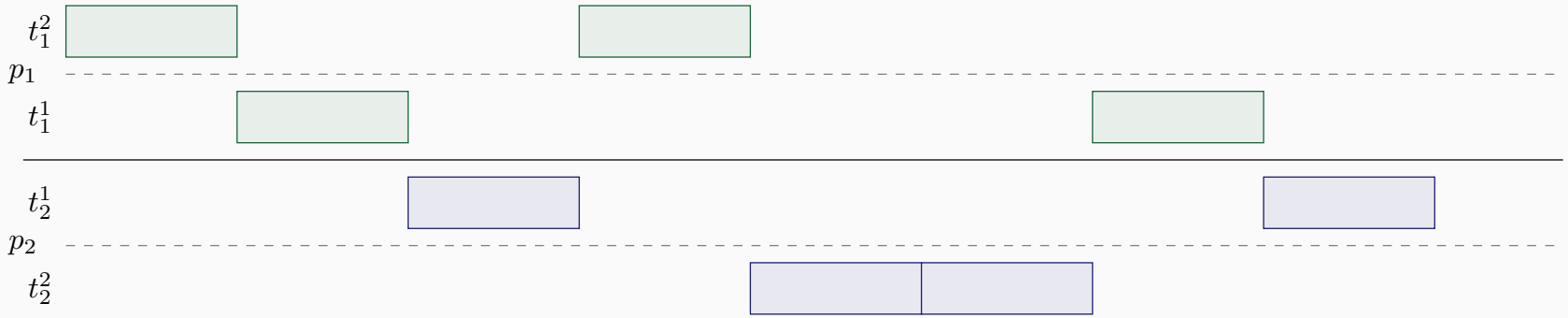
Pre formálnejší prístup viď napríklad IV010 alebo IB109.

Vlákna: Terminológia

✗ konkurentný, ✗ paralelný systém

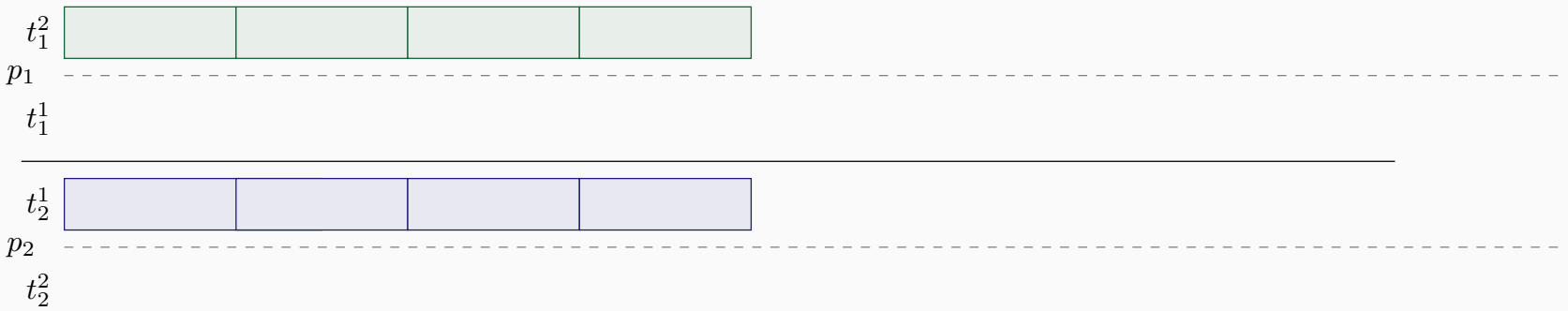


✓ konkurentný, ✗ paralelný systém



Vlákna: Terminológia

✗ konkurentný, ✓ paralelný systém



✓ konkurentný, ✓ paralelný systém

Vlákna

Vlákna zdieľajú

- obsluhu signálov
- pamäť
- premenné prostredia
- otvorené súbory
- pracovný a koreňový adresár

Súbeh

Pri prístupe ku zdieľaným zdrojom je nutná synchronizácia.
Tej sa venuje seminár 8.

Vlákna

Vlákna *ne*zdieľajú

- kontext CPU
- zásobník
- masku blokovaných signálov
- rad čakajúcich signálov
- niektoré špecifické hodnoty (napr. `errno`)

Vlákna: Native POSIX Threads Library (Linux)

- Proces je skupina vláken (*Thread Group*).
  - Hlavné vlákno sa volá *Thread Group Leader*.
  - Každé vlákno má svoje TID (*Thread ID*).
  - PID = TGID (*Thread Group ID*) = TID hlavného vlákna.
- Každé vlákno je plánované samostatne.

`clone()`

Vlákna aj procesy v Linuxe vznikajú volaním `clone()`.

Historické okienko - Linux Threads

Pôvodná implementácia vláken, viď `man 7 pthreads`.

Preklad

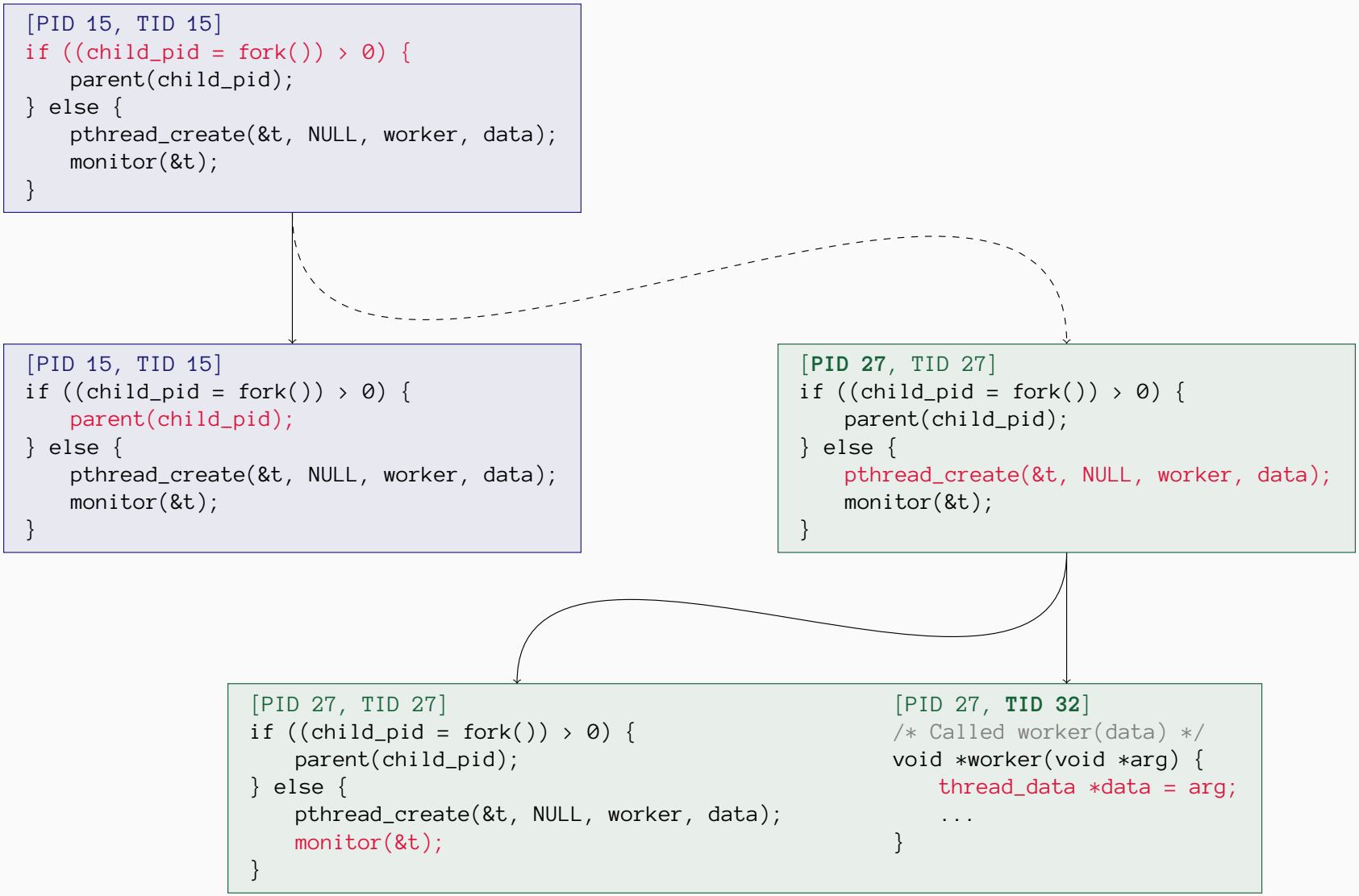
Používajte `-pthread` pre kompiláciu **aj** linkovanie.
Viď `man gcc`.

```
CFLAGS += -pthread
LDFLAGS += -pthread
```

Vlákna: Vytvorenie

```c
#include <pthread.h>
int pthread_create(pthread_t *thread, const pthread_attr_t *attr,
                   void *(*start)(void*), void *arg);
```

Vlákna: Vytvorenie



Vlákna: Chyby

Funkcie `pthread_*()` **nenastavujú** `errno`, chybový kód vrátia.

```
int pterrno;
if ((pterrno = pthread_join(thread, &rv)) != 0)
    errx(1, "pthread_join: %s", strerror(pterrno));
```

Hlásenie chýb

`man 3 error` (GNU)

Vlákna: Atribúty

Opaque type, ktorý sa nastavuje funkciami.

```
pthread_attr_t attr;
pthread_attr_init(&attr);          /* 1 Initialise */

pthread_attr_set*(&attr, …);       /* 2 Set attributes */
pthread_attr_set*(&attr, …);

pthread_create(&t, &attr, …);      /* 3 Use */

pthread_attr_destroy(&attr);       /* 4 Release */
```

Vlákna: Atribúty

Užitočné atribúty

`pthread_attr_get…`
`pthread_attr_set…`

- `…detachstate()`
- `…schedparam()`
- `…schedpolicy()`
- `…scope()`

```
man pthread_attr_setζ.
```

Vlákna: Identifikácia

POSIX

```
#include <pthread.h>
pthread_t pthread_self(void);
int pthread_equal(pthread_t t1, pthread_t t2);
```

Jedinečnosť je zaručená len v kontexte procesu.

`pthread_t` je *opaque type*!

Vlákna: Identifikácia

Linux

Vlákna majú vlastné systémové ID, *Thread ID*.
Táto identifikácia je *závislá na platforme*.

```c
/* glibc <2.30 */
#include <sys/syscall.h>
#include <unistd.h>
(pid_t) syscall(SYS_gettid);
```

```c
/* glibc ≥2.30 */
#define _GNU_SOURCE
#include <unistd.h>
pid_t gettid(void);
```

! Interpretácia `pthread_t`

`pthread_t` v Linuxe obsahuje číslo, ktoré **nie je** TID!

Vlákna: Zlúčenie

```c
#include <pthread.h>
int pthread_join(pthread_t thread, void **retval);
#define PTHREAD_CANCELED /* … */
```

- Vlákno nemôže zlúčiť samo seba.
- Každé vlákno môže byť zlúčené nanajvýš raz.

Vlákna: Zlúčenie (GNU)

⚠ GNU rozšírenia

_np znamená *non-portable*!

```
#define _GNU_SOURCE
#include <pthread.h>
int pthread_tryjoin_np(pthread_t thread, void **retval);
int pthread_timedjoin_np(pthread_t thread, void **retval,
        const struct timespec *timeout);
```

Vlákna: Odpojenie

```c
#include <pthread.h>
int pthread_detach(pthread_t thread);
```

- Vlákno môže zavolať aj na sebe.
- Na odpojené vlákno nie je možné čakať.
- `pthread_attr_setdetachstate()`

Vlákna: Ukončenie

Vlákno končí

- volaním `pthread_exit()`
- vrátením hodnoty z obslužnej funkcie
- zrušením z iného vlákna

Vlákna: Ukončenie

```c
#include <pthread.h>
void pthread_exit(void *retval);
```

- Ekvivalent `return retval;` z obslužnej funkcie.
- Po skončení posledného vlákna sa zavolá `exit()`.

! Rozdiel v hlavnom vlákne

`return` alebo `exit()` z `main()` ukončí celý **proces**.

Volanie `pthread_exit()` ponechá ostatné vlákna nažive.

Vlákna: Zrušenie

```c
#include <pthread.h>
int pthread_cancel(pthread_t thread);
#define PTHREAD_CANCELED /* … */
```

Vlákna: Zrušenie

Vlákno môže byť z pohľadu zrušenia v troch stavoch:

asynchrónne zrušiteľné
: zruší sa okamžite

synchrónne zrušiteľné
: zruší sa na povolenom mieste

nezrušiteľné
: zruší sa pri prepnutí na *zrušiteľné*

Vlákna: Zrušenie

```c
#include <pthread.h>

int pthread_setcancelstate(int state, int *oldstate);
#define PTHREAD_CANCEL_ENABLE /* … */   // Cancelable
#define PTHREAD_CANCEL_DISABLE /* … */  // Not cancelable

int pthread_setcanceltype(int type, int *olddype);
#define PTHREAD_CANCEL_ASYNCHRONOUS /* … */
#define PTHREAD_CANCEL_DEFERED /* … */
```

Body zrušenia

- implicitné
  `man 7 pthreads` → Cancellation points
- explicitné

  ```c
  void pthread_testcancel(void);
  ```

Vlákna: Deštruktory

```
void pthread_cleanup_push(void (*routine)(void*), void *arg);
void pthread_cleanup_pop(int execute);
```

Tieto symboly môžu byť makrá, ktoré vytvárajú lexikálny rozsah. Preto *musia* byť použité v páre a v tej istej funkcii.

Thread-Local Storage

Thread-Local Storage

Bežné globálne premenné sú viditeľné pre celý proces.

Thread-Local Storage: “Globálne” premenné pre vlákna.

Thread-Local Storage: POSIX kľúče

Kľúč k dátam

```
int pthread_key_create(pthread_key_t *key, void (*dtor)(void*));
int pthread_key_delete(pthread_key_t key);
```

Tento kľúč je platný vo všetkých vláknach, *vrátane* budúcich.
Musia si ho však nejak podať.

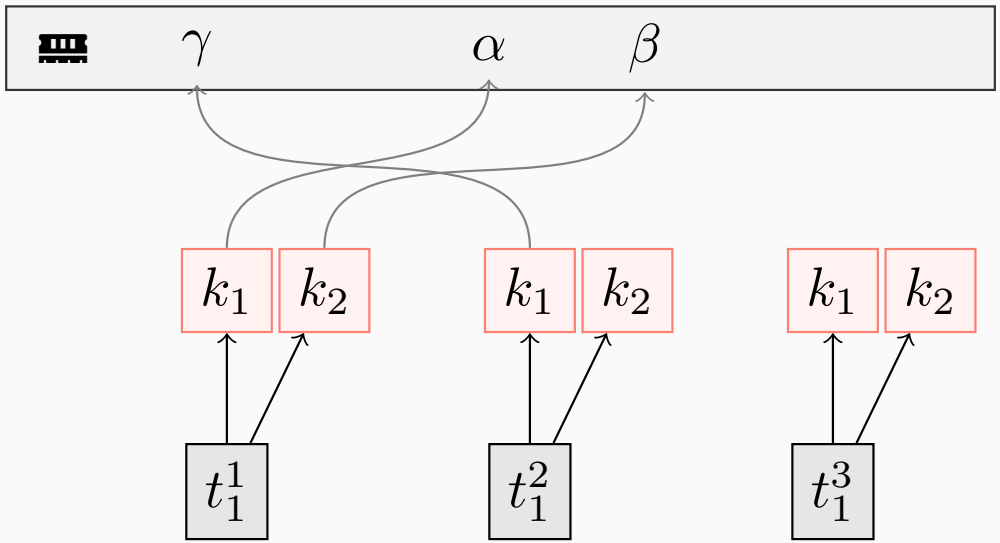
Prístup k hodnote

```
void *pthread_getspecific(pthread_key_t key);
void *pthread_setspecific(pthread_key_t key, const void *value);
```

Čo je `errno`?

Thread-Local Storage: POSIX kľúče

```
/* t₁ */ pthread_key_create(k₁, free);
         /* Create t₂ */
/* t₁ */ pthread_key_create(k₂, free);
         /* Create t₃ */
/* t₁ */ pthread_set_specific(k₁, α);
/* t₁ */ pthread_set_specific(k₂, β);
/* t₂ */ pthread_set_specific(k₁, γ);
```



Thread-Local Storage: Podpora v ELF

ELF podporuje `.tdata` a `.tbss` sekcie.

Prekladač GCC

```
__thread T var;
static __thread T var;
extern __thread T var;
```

Štandard C11

```
_Thread_local T var;
#define thread_local _Thread_local
```

Od C23 je `thread_local` plnohodnotné kľúčové slovo.

Vlákna a svet

Vlákna vs procesy

`pthread_create()`	≈ `fork()`
`pthread_exit()`	≈ `exit_group()`
`pthread_join()`	≈ `waitpid()`
`pthread_cleanup_push()`	≈ `atexit()`
`pthread_self()`	≈ `getpid()`
`pthread_cancel()`	≈ `kill()`

Vlákna a `fork()`

Problém

`fork()` do potomka skopíruje všetky zdroje,
pokračuje len volajúce vlákno.

Potenciálne uviaznutie (*deadlock*) alebo únik zdrojov.

Riešenia

1. Volajte `fork()` len pred vytvorením vláken.
2. Čo najskôr po `fork()` zavolajte `exec*()`.
3. ```
   typedef void (cb_t)(void);
   int pthread_atfork(cb_t *prepare, cb_t *parent, cb_t *child);
   ```

   Táto funkcia je však veľmi ťažkopádna.
   ```

Vlákna a signály

- **Process-Directed Signal** vs **Thread-Directed Signal**
- Vlákno si môže zamaskovať signály
- Vlákno môže poslať signál inému vláknu
  - Preferujte synchronizačné mechanizmy (ďalší seminár)

```
#include <pthread.t>
#include <signal.h>
int pthread_kill(pthread_t thread, int sig);
int pthread_sigmask(int how, const sigset_t *set, sigset_t *old);
```

`pthread_sigqueue()` (GNU)

Vlákna a štandardné C

C11

- Väčšina funkcií `pthread_α()` → `thrd_α()`
  - Pozor na občasné rozdiely v sémantike
- Thread-Local Storage: `tss_t`, `tss_*()`
- Thread-Local Types

  ```
  _Thread_local T var;
  ```

- Atomické typy

  ```
  _Atomic T var;
  ```

C23

- `_Thread_local` → `thread_local`

Záver

Záver

- Concurrency Support Library (cppreference)
- POSIX Thread Libraries
- ELF Handling for Thread-Local Storage
- Why Threads Are a Bad Idea (for most purposes)