Zaawansowane programowanie w C++

Michał Tanaś, PhD

Adam Mickiewicz University, Faculty of Physics

http://www.amu.edu.pl/~mtanas

Michal.Tanas@amu.edu.pl

Programy wielowątkowe – PThreads

PThreads (libpthread – POSIX Threads Library)

- Wspólne dla wszystkich Unixów API umożliwiające programowanie wielowątkowe w C/C++
- Zdefiniowane jako standard IEEE POSIX 1003.1c (1995)
- Uważane za najwydajniejszą sposób tworzenia programów wielowątkowych dla Unixów.
- Istnieje wersja PThreads dla Windows.

Programy wielowątkowe – PThreads

PThreads (libpthread – POSIX Threads Library)

- Wspólne dla wszystkich Unixów API umożliwiające programowanie wielowątkowe w C/C++
- Zdefiniowane jako standard IEEE POSIX 1003.1c (1995)
- Uważane za najwydajniejszą sposób tworzenia programów wielowątkowych dla Unixów.
- Istnieje wersja PThreads dla Windows.

Programy wielowątkowe – PThreads

PThreads (libpthread – POSIX Threads Library)

- Wspólne dla wszystkich Unixów API umożliwiające programowanie wielowątkowe w C/C++
- Zdefiniowane jako standard IEEE POSIX 1003.1c (1995)
- Uważane za najwydajniejszą sposób tworzenia programów wielowątkowych dla Unixów.
- Istnieje wersja PThreads dla Windows.

Programy wielowątkowe – PThreads

PThreads (libpthread – POSIX Threads Library)

- Wspólne dla wszystkich Unixów API umożliwiające programowanie wielowątkowe w C/C++
- Zdefiniowane jako standard IEEE POSIX 1003.1c (1995)
- Uważane za najwydajniejszą sposób tworzenia programów wielowątkowych dla Unixów.
- Istnieje wersja PThreads dla Windows.

Programy wielowątkowe – PThreads

Użycie funkcji z biblioteki PThreads wymaga dołączenia nagłóka pthread.h.

Wątek w rozumieniu PThreads: procedura wykonywana równolegle z resztą programu.

Programy wielowątkowe – PThreads

Użycie funkcji z biblioteki PThreads wymaga dołączenia nagłóka pthread.h.

Wątek w rozumieniu PThreads: procedura wykonywana równolegle z resztą programu.

Programy wielowątkowe – PThreads

Wątki tworzone przez PThread współdzielą

- przestrzeń adresową
- zmienne (działają standardowe ograniczenia widoczności zmiennych)
- deskryptory plików
- procedury obsługi sygnałów (ale nie reakcje na sygnały!)
- katalog roboczy
- UID i GID

Programy wielowątkowe – PThreads

Wątki tworzone przez PThread współdzielą

- przestrzeń adresową
- zmienne (działają standardowe ograniczenia widoczności zmiennych)
- deskryptory plików
- procedury obsługi sygnałów (ale nie reakcje na sygnały!)
- katalog roboczy
- UID i GID

Programy wielowątkowe – PThreads

Wątki tworzone przez PThread współdzielą

- przestrzeń adresową
- zmienne (działają standardowe ograniczenia widoczności zmiennych)
- deskryptory plików
- procedury obsługi sygnałów (ale nie reakcje na sygnały!)
- katalog roboczy
- UID i GID

Programy wielowątkowe – PThreads

Wątki tworzone przez PThread współdzielą

- przestrzeń adresową
- zmienne (działają standardowe ograniczenia widoczności zmiennych)
- deskryptory plików
- procedury obsługi sygnałów (ale nie reakcje na sygnały!)
- katalog roboczy
- UID i GID

Programy wielowątkowe – PThreads

Wątki tworzone przez PThread współdzielą

- przestrzeń adresową
- zmienne (działają standardowe ograniczenia widoczności zmiennych)
- deskryptory plików
- procedury obsługi sygnałów (ale nie reakcje na sygnały!)
- katalog roboczy
- UID i GID

Programy wielowątkowe – PThreads

Wątki tworzone przez PThread współdzielą

- przestrzeń adresową
- zmienne (działają standardowe ograniczenia widoczności zmiennych)
- deskryptory plików
- procedury obsługi sygnałów (ale nie reakcje na sygnały!)
- katalog roboczy
- UID i GID

Programy wielowątkowe – PThreads

Wątki tworzone przez PThread nie współdzielą

- stosu
- rejestrów
- reakcji na sygnały (sigmask)
- zmiennych lokalnych

Programy wielowątkowe – PThreads

Wątki tworzone przez PThread nie współdzielą

- stosu
- rejestrów
- reakcji na sygnały (sigmask)
- zmiennych lokalnych

Programy wielowątkowe – PThreads

Wątki tworzone przez PThread nie współdzielą

- stosu
- rejestrów
- reakcji na sygnały (sigmask)
- zmiennych lokalnych

Programy wielowątkowe – PThreads

Wątki tworzone przez PThread nie współdzielą

- stosu
- rejestrów
- reakcji na sygnały (sigmask)
- zmiennych lokalnych

Programy wielowątkowe – PThreads

```
int pthread_create(pthread_t *th,
    pthread_attr_t *attr,
    void * (*fn)(void *), void *arg);
```

- tworzy wątek potomny
- w kontekście procesu macierzystego zwraca PID wątku (procesu) potomnego
- w zmiennej th zapisuje informacje utworzonym wątku

Programy wielowątkowe – PThreads

```
int pthread_create(pthread_t *th,
    pthread_attr_t *attr,
    void * (*fn)(void *), void *arg);
```

- tworzy wątek potomny
- w kontekście procesu macierzystego zwraca PID wątku (procesu) potomnego
- w zmiennej th zapisuje informacje utworzonym wątku

Programy wielowątkowe – PThreads

```
int pthread_create(pthread_t *th,
    pthread_attr_t *attr,
    void * (*fn)(void *), void *arg);
```

- tworzy wątek potomny
- w kontekście procesu macierzystego zwraca PID wątku (procesu) potomnego
- w zmiennej th zapisuje informacje utworzonym wątku

Programy wielowątkowe – PThreads

```
int pthread_create(pthread_t *th,
    pthread_attr_t *attr,
    void * (*fn)(void *), void *arg);
```

- w kontekście procesu potomnego wywoływana jest funkcja fn z argumentem arg.
- wyjście z funkcji fn w kontekście wątku potomnego jest równoznaczne z zakończeniem tego wątku.
- wartość zwracana przez funkcję fn jest kodem wyjścia wątku potomnego

Programy wielowątkowe – PThreads

```
int pthread_create(pthread_t *th,
    pthread_attr_t *attr,
    void * (*fn)(void *), void *arg);
```

- w kontekście procesu potomnego wywoływana jest funkcja fn z argumentem arg.
- wyjście z funkcji fn w kontekście wątku potomnego jest równoznaczne z zakończeniem tego wątku.
- wartość zwracana przez funkcję fn jest kodem wyjścia wątku potomnego

Programy wielowątkowe – PThreads

```
int pthread_create(pthread_t *th,
    pthread_attr_t *attr,
    void * (*fn)(void *), void *arg);
```

- w kontekście procesu potomnego wywoływana jest funkcja fn z argumentem arg.
- wyjście z funkcji fn w kontekście wątku potomnego jest równoznaczne z zakończeniem tego wątku.
- wartość zwracana przez funkcję fn jest kodem wyjścia wątku potomnego

Programy wielowątkowe – PThreads

```
int pthread_create(pthread_t *th,
    pthread_attr_t *attr,
    void * (*fn)(void *), void *arg);
```

Zmienna attr

- określa atrybuty wątku
- musi być wcześniej zainicjalizowana przy pomocy funkcji
  `int pthread_init_attr(pthread_attr_t *attr);`
- zmiany atrybutów dokonuje się przy pomocy rodziny funkcji
  `int pthread_attr_set`*nazwa*`(pthread_attr_t *attr, int wartość);`

Programy wielowątkowe – PThreads

```
int pthread_create(pthread_t *th,
    pthread_attr_t *attr,
    void * (*fn)(void *), void *arg);
```

Zmienna attr

- określa atrybuty wątku
- musi być wcześniej zainicjalizowana przy pomocy funkcji
  `int pthread_init_attr(pthread_attr_t *attr);`
- zmiany atrybutów dokonuje się przy pomocy rodziny funkcji
  `int pthread_attr_set`*nazwa*`(pthread_attr_t *attr, int wartość);`

Programy wielowątkowe – PThreads

```
int pthread_create(pthread_t *th,
    pthread_attr_t *attr,
    void * (*fn)(void *), void *arg);
```

Zmienna attr

- określa atrybuty wątku
- musi być wcześniej zainicjalizowana przy pomocy funkcji
  `int pthread_init_attr(pthread_attr_t *attr);`
- zmiany atrybutów dokonuje się przy pomocy rodziny funkcji
  `int pthread_attr_set`*nazwa*`(pthread_attr_t *attr, int wartość);`

Programy wielowątkowe – PThreads

Atrybuty wątków

- detachstate – możliwość synchronizowania tworzonego wątku z innymi wątkami.
  - PTHREAD_CREATE_JOINABLE – inne wątki mogą oczekiwać na zakończenie tworzonego wątku, tworzony wątek po zakończeniu utworzy zombiego. Opcja domyślna.
  - PTHREAD_CREATE_DETACHED – inne wątki nie mogą oczekiwać na zakończenie tworzonego wątku, tworzony wątek po zakończeniu natychmiast zwalnia wszystkie zasoby (nie tworzy zombiego).

Programy wielowątkowe – PThreads

Atrybuty wątków

- detachstate – możliwość synchronizowania tworzonego wątku z innymi wątkami.
  - PTHREAD_CREATE_JOINABLE – inne wątki mogą oczekiwać na zakończenie tworzonego wątku, tworzony wątek po zakończeniu utworzy zombiego. Opcja domyślna.
  - PTHREAD_CREATE_DETACHED – inne wątki nie mogą oczekiwać na zakończenie tworzonego wątku, tworzony wątek po zakończeniu natychmiast zwalnia wszystkie zasoby (nie tworzy zombiego).

Programy wielowątkowe – PThreads

Atrybuty wątków

- detachstate – możliwość synchronizowania tworzonego wątku z innymi wątkami.
  - PTHREAD_CREATE_JOINABLE – inne wątki mogą oczekiwać na zakończenie tworzonego wątku, tworzony wątek po zakończeniu utworzy zombiego. Opcja domyślna.
  - PTHREAD_CREATE_DETACHED – inne wątki nie mogą oczekiwać na zakończenie tworzonego wątku, tworzony wątek po zakończeniu natychmiast zwalnia wszystkie zasoby (nie tworzy zombiego).

Programy wielowątkowe – PThreads

Atrybuty wątków

- schedpolicy – tworzenie wątków czasu rzeczywistego.
  - SCHED_OTHER – wątek standardowy (nie czasu rzeczywistego). Opcja domyślna.
  - SCHED_RR – wątek czasu rzeczywistego szeregowany „round robin”. Wymaga uprawnień roota.
  - SCHED_FIFO – wątek czasu rzeczywistego szeregowany „first in first out”. Wymaga uprawnień roota.

  Przy pomocy funkcji pthread_attr_setschedparam można określić priorytety wątków czasu rzeczywistego.

Programy wielowątkowe – PThreads

Atrybuty wątków

- schedpolicy – tworzenie wątków czasu rzeczywistego.
  - SCHED_OTHER – wątek standardowy (nie czasu rzeczywistego). Opcja domyślna.
  - SCHED_RR – wątek czasu rzeczywistego szeregowany „round robin”. Wymaga uprawnień roota.
  - SCHED_FIFO – wątek czasu rzeczywistego szeregowany „first in first out”. Wymaga uprawnień roota.

  Przy pomocy funkcji pthread_attr_setschedparam można określić priorytety wątków czasu rzeczywistego.

Programy wielowątkowe – PThreads

Atrybuty wątków

- schedpolicy – tworzenie wątków czasu rzeczywistego.
  - SCHED_OTHER – wątek standardowy (nie czasu rzeczywistego). Opcja domyślna.
  - SCHED_RR – wątek czasu rzeczywistego szeregowany „round robin”. Wymaga uprawnień roota.
  - SCHED_FIFO – wątek czasu rzeczywistego szeregowany „first in first out”. Wymaga uprawnień roota.

Przy pomocy funkcji pthread_attr_setschedparam można określić priorytety wątków czasu rzeczywistego.

Programy wielowątkowe – PThreads

Atrybuty wątków

- schedpolicy – tworzenie wątków czasu rzeczywistego.
  - SCHED_OTHER – wątek standardowy (nie czasu rzeczywistego). Opcja domyślna.
  - SCHED_RR – wątek czasu rzeczywistego szeregowany „round robin”. Wymaga uprawnień roota.
  - SCHED_FIFO – wątek czasu rzeczywistego szeregowany „first in first out”. Wymaga uprawnień roota.

Przy pomocy funkcji pthread_attr_setschedparam można określić priorytety wątków czasu rzeczywistego.

Programy wielowątkowe – PThreads

Atrybuty wątków

- schedpolicy – tworzenie wątków czasu rzeczywistego.
  - SCHED_OTHER – wątek standardowy (nie czasu rzeczywistego). Opcja domyślna.
  - SCHED_RR – wątek czasu rzeczywistego szeregowany „round robin”. Wymaga uprawnień roota.
  - SCHED_FIFO – wątek czasu rzeczywistego szeregowany „first in first out”. Wymaga uprawnień roota.

Przy pomocy funkcji pthread_attr_setschedparam można określić priorytety wątków czasu rzeczywistego.

Programy wielowątkowe – PThreads

Atrybuty wątków

- inheritsched – umożliwia dziedziczenie schedpolicy i schedparam
  - PTHREAD_INHERIT_SCHED – tworzony wątek będzie miał identyczne schedpolicy i schedparam jak wątek macierzysty.
  - PTHREAD_EXPLICIT_SCHED – tworzony wątek będzie miał schedpolicy i schedparam zgodne z atrybutami przekazanymi do pthread_create.

Programy wielowątkowe – PThreads

Atrybuty wątków

- inheritsched – umożliwia dziedziczenie schedpolicy i schedparam
  - PTHREAD_INHERIT_SCHED – tworzony wątek będzie miał identyczne schedpolicy i schedparam jak wątek macierzysty.
  - PTHREAD_EXPLICIT_SCHED – tworzony wątek będzie miał schedpolicy i schedparam zgodne z atrybutami przekazanymi do pthread_create.

Programy wielowątkowe – PThreads

Atrybuty wątków

- inheritsched – umożliwia dziedziczenie schedpolicy i schedparam
  - PTHREAD_INHERIT_SCHED – tworzony wątek będzie miał identyczne schedpolicy i schedparam jak wątek macierzysty.
  - PTHREAD_EXPLICIT_SCHED – tworzony wątek będzie miał schedpolicy i schedparam zgodne z atrybutami przekazanymi do pthread_create.

Programy wielowątkowe – PThreads

Atrybuty wątków

- scope – decyduje względem czego liczone będą priorytety wątków
  - PTHREAD_SCOPE_SYSTEM – priorytety wątków liczone są globalnie w całym systemie. Opcja domyślna (i pod Linuxem jedyna możliwa).
  - PTHREAD_SCOPE_PROCESS – priorytety wątków liczone są względem procesu macierzystego. Nie zaimplementowane w Linuxie.

Programy wielowątkowe – PThreads

Atrybuty wątków

- scope – decyduje względem czego liczone będą priorytety wątków
  - PTHREAD_SCOPE_SYSTEM – priorytety wątków liczone są globalnie w całym systemie. Opcja domyślna (i pod Linuxem jedyna możliwa).
  - PTHREAD_SCOPE_PROCESS – priorytety wątków liczone są względem procesu macierzystego. Nie zaimplementowane w Linuxie.

Programy wielowątkowe – PThreads

Atrybuty wątków

- scope – decyduje względem czego liczone będą priorytety wątków
  - PTHREAD_SCOPE_SYSTEM – priorytety wątków liczone są globalnie w całym systemie. Opcja domyślna (i pod Linuxem jedyna możliwa).
  - PTHREAD_SCOPE_PROCESS – priorytety wątków liczone są względem procesu macierzystego. Nie zaimplementowane w Linuxie.

Programy wielowątkowe – PThreads

```
void pthread_exit(void *retval);
```

Natychmiast kończy wykonywanie wątku w którym została wywołana. Odpowiednik funkcji exit.

Programy wielowątkowe – PThreads

```
int pthread_join(pthread_t th,
    void **thread_return);
```

Zatrzymuje wątek w którym została wywołana w oczekiwaniu na zakończenie wątku th.

- usuwa zombiego wątku th
- wątek th nie może być DETACHED
- odpowiednik funkcji waitpid
- wątek nie może czekać sam na siebie ;-)

Programy wielowątkowe – PThreads

```
int pthread_join(pthread_t th,
    void **thread_return);
```

Zatrzymuje wątek w którym została wywołana w oczekiwaniu na zakończenie wątku th.

- usuwa zombiego wątku th
- wątek th nie może być DETACHED
- odpowiednik funkcji waitpid
- wątek nie może czekać sam na siebie ;-)

Programy wielowątkowe – PThreads

```
int pthread_join(pthread_t th,
    void **thread_return);
```

Zatrzymuje wątek w którym została wywołana w oczekiwaniu na zakończenie wątku th.

- usuwa zombiego wątku th
- wątek th nie może być DETACHED
- odpowiednik funkcji waitpid
- wątek nie może czekać sam na siebie ;-)

Programy wielowątkowe – PThreads

```
int pthread_join(pthread_t th,
    void **thread_return);
```

Zatrzymuje wątek w którym została wywołana w oczekiwaniu na zakończenie wątku th.

- usuwa zombiego wątku th
- wątek th nie może być DETACHED
- odpowiednik funkcji waitpid
- wątek nie może czekać sam na siebie ;-)

Programy wielowątkowe – PThreads

```
int pthread_detach(pthread_t th);
```

Zmienia możliwość synchronizacji istniejącego wątku na PTHREAD_CREATE_DETACHED. Wątek DETACHED:

- natychmiast po zakończeniu zwalnia wszystkie swoje zasoby
- nigdy nie tworzy zombiego
- inne wątki nie mogą się synchronizować z wątkiem DETACHED przy pomocy pthread_join

  Uwaga - jeżeli jakieś wątki już oczekują na zakończenie wątku th funkcja pthread_detach nic nie robi.

Programy wielowątkowe – PThreads

```
int pthread_detach(pthread_t th);
```

Zmienia możliwość synchronizacji istniejącego wątku na PTHREAD_CREATE_DETACHED. Wątek DETACHED:

- natychmiast po zakończeniu zwalnia wszystkie swoje zasoby
- nigdy nie tworzy zombiego
- inne wątki nie mogą się synchronizować z wątkiem DETACHED przy pomocy pthread_join

  Uwaga - jeżeli jakieś wątki już oczekują na zakończenie wątku th funkcja pthread_detach nic nie robi.

Programy wielowątkowe – PThreads

int pthread_detach(pthread_t th);

Zmienia możliwość synchronizacji istniejącego wątku na PTHREAD_CREATE_DETACHED. Wątek DETACHED:

- natychmiast po zakończeniu zwalnia wszystkie swoje zasoby
- nigdy nie tworzy zombiego
- inne wątki nie mogą się synchronizować z wątkiem DETACHED przy pomocy pthread_join

  Uwaga - jeżeli jakieś wątki już oczekują na zakończenie wątku th funkcja pthread_detach nic nie robi.

Programy wielowątkowe – PThreads

```
int pthread_detach(pthread_t th);
```

Zmienia możliwość synchronizacji istniejącego wątku na PTHREAD_CREATE_DETACHED. Wątek DETACHED:

- natychmiast po zakończeniu zwalnia wszystkie swoje zasoby
- nigdy nie tworzy zombiego
- inne wątki nie mogą się synchronizować z wątkiem DETACHED przy pomocy pthread_join

  Uwaga - jeżeli jakieś wątki już oczekują na zakończenie wątku th funkcja pthread_detach nic nie robi.

Programy wielowątkowe – PThreads

MutEx = Mutual Exclusion

mutex – struktura używana do zagwarantowania wątkowi WYŁĄCZNEGO dostępu do współdzielonych danych.

Stany mutexu:

- odblokowany
- zablokowany (posiadany) przez jeden wątek

Programy wielowątkowe – PThreads

MutEx = Mutual Exclusion

mutex – struktura używana do zagwarantowania wątkowi WYŁĄCZNEGO dostępu do współdzielonych danych.

Stany mutexu:

- odblokowany
- zablokowany (posiadany) przez jeden wątek

Programy wielowątkowe – PThreads

MutEx = Mutual Exclusion

mutex – zasada działania mutexu:

- Nigdy dwa wątki nie mogą posiadać równocześnie blokady tego samego mutexu.
- Wykonywanie wątku który chce założyć blokadę na mutex już zablokowany przez drugi wątek jest zatrzymywane do czasu zwolnienia blokady przez drugi wątek.

Programy wielowątkowe – PThreads

MutEx = Mutual Exclusion

mutex – zasada działania mutexu:

- Nigdy dwa wątki nie mogą posiadać równocześnie blokady tego samego mutexu.
- Wykonywanie wątku który chce założyć blokadę na mutex już zablokowany przez drugi wątek jest zatrzymywane do czasu zwolnienia blokady przez drugi wątek.

Programy wielowątkowe – PThreads

Rodzaje mutexów (uwaga - tylko Linux):

- FAST – brak sprawdzania wielokrotnych blokad, próba nałożenia kolejnej blokady na mutex już zablokowany przez dany wątek powoduje zakleszczenie tego wątku. Typ domyślny.

- ERRORCHECK – próba nałożenia kolejnej blokady na mutex już zablokowany przez dany wątek nie powiedzie się, wykonywanie wątku nie zostanie zatrzymane.
- RECURSIVE – wątek może nałożyć wiele blokad na ten sam mutex. Odblokowanie mutexu następuje po zwolnieniu wszystkich blokad.

Programy wielowątkowe – PThreads

Rodzaje mutexów (uwaga - tylko Linux):

- FAST – brak sprawdzania wielokrotnych blokad, próba nałożenia kolejnej blokady na mutex już zablokowany przez dany wątek powoduje zakleszczenie tego wątku. Typ domyślny.

- ERRORCHECK – próba nałożenia kolejnej blokady na mutex już zablokowany przez dany wątek nie powiedzie się, wykonywanie wątku nie zostanie zatrzymane.
- RECURSIVE – wątek może nałożyć wiele blokad na ten sam mutex. Odblokowanie mutexu następuje po zwolnieniu wszystkich blokad.

Programy wielowątkowe – PThreads

Rodzaje mutexów (uwaga - tylko Linux):

- FAST – brak sprawdzania wielokrotnych blokad, próba nałożenia kolejnej blokady na mutex już zablokowany przez dany wątek powoduje zakleszczenie tego wątku. Typ domyślny.

- ERRORCHECK – próba nałożenia kolejnej blokady na mutex już zablokowany przez dany wątek nie powiedzie się, wykonywanie wątku nie zostanie zatrzymane.

- RECURSIVE – wątek może nałożyć wiele blokad na ten sam mutex. Odblokowanie mutexu następuje po zwolnieniu wszystkich blokad.

Programy wielowątkowe – PThreads

```
int pthread_mutexattr_init(
      pthread_mutexattr_t *attr);
```

Tworzy strukturę reprezentującą atrybuty mutexu i nadaje tym atrybutom standardowe wartości. W Linuxie jedynym atrybutem mutexu jest jego rodzaj, domyślnie: FAST

Programy wielowątkowe – PThreads

```
int pthread_mutexattr_settype(
      pthread_mutexattr_t *attr, int kind);
```

Zmienia typ mutexu:

- PTHREAD_MUTEX_FAST_NP – mutex typu FAST
- PTHREAD_MUTEX_ERRORCHECK_NP – mutex typu ERRORCHECK
- PTHREAD_MUTEX_RECURSIVE_NP – mutex typu RECURSIVE

Programy wielowątkowe – PThreads

```
int pthread_mutexattr_settype(
      pthread_mutexattr_t *attr, int kind);
```

Zmienia typ mutexu:

- PTHREAD_MUTEX_FAST_NP – mutex typu FAST
- PTHREAD_MUTEX_ERRORCHECK_NP – mutex typu ERRORCHECK
- PTHREAD_MUTEX_RECURSIVE_NP – mutex typu RECURSIVE

Programy wielowątkowe – PThreads

```
int pthread_mutexattr_settype(
      pthread_mutexattr_t *attr, int kind);
```

Zmienia typ mutexu:

- PTHREAD_MUTEX_FAST_NP – mutex typu FAST
- PTHREAD_MUTEX_ERRORCHECK_NP – mutex typu ERRORCHECK
- PTHREAD_MUTEX_RECURSIVE_NP – mutex typu RECURSIVE

Programy wielowątkowe – PThreads

```
int pthread_mutex_init(
      pthread_mutex_t *mutex,
      pthread_mutexattr_t *mutexattr);
```

Tworzy mutex i nadaje mu podane atrybuty.

Programy wielowątkowe – PThreads

```
int pthread_mutex_lock(pthread_mutex_t *mutex);
```

Zakłada blokadę mutexu mutex na rzecz wątku z którego została wywołana.

- Jeżeli mutex nie jest w danej chwili zablokowany przez inny wątek staje się natychmiast zablokowany przez wątek bieżący. Następuje powrót z funkcji pthread_mutex_lock.
- Jeżeli mutex jest już zablokowany przez inny wątek funkcja pthread_mutex_lock wstrzymuje wykonywanie bieżącego wątku do czasu aż blokada ta zostanie zwolniona. Wtedy na mutex zostanie nałożona blokada na rzecz bieżącego wątku i nastąpi powrót z funkcji pthread_mutex_lock.

Programy wielowątkowe – PThreads

```
int pthread_mutex_lock(pthread_mutex_t *mutex);
```

Zakłada blokadę mutexu mutex na rzecz wątku z którego została wywołana.

- Jeżeli mutex nie jest w danej chwili zablokowany przez inny wątek staje się natychmiast zablokowany przez wątek bieżący. Następuje powrót z funkcji pthread_mutex_lock.
- Jeżeli mutex jest już zablokowany przez inny wątek funkcja pthread_mutex_lock wstrzymuje wykonywanie bieżącego wątku do czasu aż blokada ta zostanie zwolniona. Wtedy na mutex zostanie nałożona blokada na rzecz bieżącego wątku i nastąpi powrót z funkcji pthread_mutex_lock.

Programy wielowątkowe – PThreads

int pthread_mutex_trylock(pthread_mutex_t *mute

Sprawdza czy można założyć blokadę mutexu mutex na rzecz wątku z którego została wywołana i jeśli tak zakłada ją.

- Jeżeli mutex nie jest w danej chwili zablokowany przez inny wątek staje się natychmiast zablokowany przez wątek bieżący (działa identycznie jak pthread_mutex_lock).
- Jeżeli mutex jest już zablokowany przez inny wątek następuje natychmiastowy powrót z funkcji ze zgłoszeniem błędu errno==EBUSY. Blokada nie jest zakładana, wątek bieżący nigdy nie jest zatrzymywany.

Programy wielowątkowe – PThreads

int pthread_mutex_trylock(pthread_mutex_t *mute

Sprawdza czy można założyć blokadę mutexu mutex na rzecz wątku z którego została wywołana i jeśli tak zakłada ją.

- Jeżeli mutex nie jest w danej chwili zablokowany przez inny wątek staje się natychmiast zablokowany przez wątek bieżący (działa identycznie jak pthread_mutex_lock).
- Jeżeli mutex jest już zablokowany przez inny wątek następuje natychmiastowy powrót z funkcji ze zgłoszeniem błędu errno==EBUSY. Blokada nie jest zakładana, wątek bieżący nigdy nie jest zatrzymywany.

Programy wielowątkowe – PThreads

```
int pthread_mutex_unlock(
      pthread_mutex_t *mutex);
```

Zdejmuje z mutexu blokadę nałożoną przez bieżący wątek.

wywołana i jeśli tak zakłada ją.

- Dla mutexów FAST i ERRORCHECK następuje natychmiastowe odblokowanie mutexu.
- Dla mutexów RECURSIVE następuje zdjęcie jednej blokady. Mutex uważa się za odblokowany dopiero po zdjęciu wszystkich blokad.

Programy wielowątkowe – PThreads

```
int pthread_mutex_unlock(
      pthread_mutex_t *mutex);
```

Zdejmuje z mutexu blokadę nałożoną przez bieżący wątek.

wywołana i jeśli tak zakłada ją.

- Dla mutexów FAST i ERRORCHECK następuje natychmiastowe odblokowanie mutexu.
- Dla mutexów RECURSIVE następuje zdjęcie jednej blokady. Mutex uważa się za odblokowany dopiero po zdjęciu wszystkich blokad.

Programy wielowątkowe – PThreads

```
int pthread_mutex_unlock(
      pthread_mutex_t *mutex);
```

Jeżeli w momencie wywołania pthread_mutex_unlock mutex jest zablokowany przez INNY wątek:

- Dla mutexów ERRORCHECK funkcja zgłasza błąd errno==EPERM. Blokada nie zostanie zdjęta.
- Dla mutexów FAST i RECURSIVE nastąpi zdjęcie blokady założonej przez inny wątek. Uwaga – zachowanie nieprzenośne i niegwarantowane, nigdy nie używać.

Programy wielowątkowe – PThreads

```
int pthread_mutex_unlock(
      pthread_mutex_t *mutex);
```

Jeżeli w momencie wywołania pthread_mutex_unlock mutex jest zablokowany przez INNY wątek:

- Dla mutexów ERRORCHECK funkcja zgłasza błąd errno==EPERM. Blokada nie zostanie zdjęta.
- Dla mutexów FAST i RECURSIVE nastąpi zdjęcie blokady założonej przez inny wątek. Uwaga – zachowanie nieprzenośne i niegwarantowane, nigdy nie używać.