Systemy operacyjne

Laboratorium 9

Mateusz Małek
19 maja 2017

Laboratorium 9

Wątki - mechanizmy synchronizacji

Tworzenie/usuwanie muteksu

- int pthread_mutex_init(pthread_mutex_t *restrict mutex, const pthread_mutexattr_t *restrict attr)
  - Inicjalizuje zmienną wskazaną przez mutex do wykorzystania w charakterze muteksu
  - Opcjonalnie można podać atrybuty muteksu w zmiennej wskazanej przez attr (jeśli chcemy pozostawić wartości domyślne, możemy jako argument attr podać NULL)
- pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;
  - Makro tworzące mutex z domyślnymi atrybutami
- int pthread_mutex_destroy(pthread_mutex_t *mutex)
  - Sprząta zasoby związane ze wskazanym muteksem
  - Funkcję należy stosować zarówno jeśli mutex zainicjalizowano funkcją pthread_mutex_init, jak i makrem PTHREAD_MUTEX_INITIALIZER
- Wszystkie powyższe konstrukcje opisane są w man 3 pthread_mutex_destroy

Atrybuty muteksu

- Zmienna typu pthread_mutexattr_t
- int pthread_mutexattr_init(pthread_mutexattr_t *attr)
  - Inicjalizuje wskazaną zmienną attr do wykorzystania jako atrybuty muteksu
- int pthread_mutexattr_destroy(pthread_mutexattr_t *attr)
  - Sprząta zasoby które zaalokowano w związku z zainicjalizowaniem zmiennej wskazanej przez attr do wykorzystania jako atrybuty muteksu
- Obie powyższe funkcje opisane w man 3 pthread_mutexattr_destroy

Atrybuty muteksu - dzielenie przez pam. wspólną

- int pthread_mutexattr_getpshared(const pthread_mutexattr_t *restrict attr, int *restrict pshared)
- int pthread_mutexattr_setpshared(pthread_mutexattr_t *attr, int pshared)
- Stałe dla argumentu pshared:
  - PTHREAD_PROCESS_SHARED - mutex może być wykorzystywany przez wszystkie wątki ze wszystkich procesów mających dostęp do obszaru pamięci wspólnej w którym umieszczono mutex
  - **PTHREAD_PROCESS_PRIVATE** - mutex może być wykorzystywany wyłącznie w ramach wątków należących do pojedynczego procesu (domyślne)
- Obie funkcje opisane w man 3 pthread_mutexattr_getpshared

Atrybuty muteksu - rodzaj

- int pthread_mutexattr_gettype(const pthread_mutexattr_t *restrict attr, int *restrict type)
- int pthread_mutexattr_settype(pthread_mutexattr_t *attr, int type)
- Stałe dla argumentu type:
  - PTHREAD_MUTEX_NORMAL - gwarantują brak sprawdzania dopuszczalności operacji lock/unlock które chcemy wykonać (np. zwolnienie muteksu którego nie zajęliśmy)
  - PTHREAD_MUTEX_ERRORCHECK - nie pozwala zająć w ramach wątku tego samego muteksu wielokrotnie lub zwolnić muteksu którego wcześniej nie zajęliśmy
  - PTHREAD_MUTEX_RECURSIVE - może zostać wielokrotnie zajęty przez ten sam wątek (musi być tyle samo razy zwolniony, żeby stał się dostępny dla innych wątków)
  - **PTHREAD_MUTEX_DEFAULT** - może być dowolnym z powyższych lub zachowywać się inaczej (domyślne)
- Obie funkcje opisane w man 3 pthread_mutexattr_gettype

Atrybuty muteksu - robustness

- int pthread_mutexattr_getrobust(const pthread_mutexattr_t *restrict attr, int *restrict robust)
- int pthread_mutexattr_setrobust(pthread_mutexattr_t *attr, int robust)
- Stałe dla argumentu robus:
  - **PTHREAD_MUTEX_STALLED** - jeśli wątek będący obecnym właścicielem zakończy się bez zwalniania muteksu, nic w związku z tym nie robimy (domyślne)
  - PTHREAD_MUTEX_ROBUST - jeśli wątek będący obecnym właścicielem zakończy się bez zwalniania muteksu, kolejny wątek który będzie chciał go zająć otrzyma błąd EOWNERDEAD i będzie mógł przywrócić mutex do stanu używalności wywołaniem funkcji int pthread_mutex_consistent(pthread_mutex_t *mutex) - man 3 pthread_mutex_consistent (Wywołanie pthread_mutex_unlock bez wcześniejszego pthread_mutex_consistent popsuje ten mutex już ostatecznie - będzie można zrobić tylko pthread_mutex_destroy)
- Obie funkcje opisane w man 3 pthread_mutexattr_getrobust

Zajmowanie/zwalnianie muteksu

- int pthread_mutex_lock(pthread_mutex_t *mutex)
  - Zajmuje mutex lub usypia wątek, jeśli zajęcie muteksu nie było możliwe
- int pthread_mutex_trylock(pthread_mutex_t *mutex)
  - Zajmuje mutex lub natychmiast zwraca błąd, jeśli zajęcie muteksu nie było możliwe
- int pthread_mutex_unlock(pthread_mutex_t *mutex)
  - Zwalnia mutex
- Wszystkie powyższe funkcje opisane są w man 3 pthread_mutex_lock
- int pthread_mutex_timedlock(pthread_mutex_t *restrict mutex, const struct timespec *restrict abstime) - man 3 pthread_mutext_timedlock
  - Działa analogicznie do pthread_mutex_lock, ale powoduje zwrócenie ETIMEDOUT jeśli nie udało się zająć muteksu przez abstime

Zachowanie muteksów a.k.a jak strzelić sobie w stopę

Rodzaj muteksu	**Robustness**	**Ponowna blokada**	**Odblokowanie gdy nie jest się właścicielem**
NORMAL	non-robust	zakleszczenie	nieokreślone zach.
NORMAL	robust	zakleszczenie	zwrócenie błędu
ERRORCHECK	obojętne	zwrócenie błędu	zwrócenie błędu
RECURSIVE	obojętne	zagłębienie	zwrócenie błędu
DEFAULT	non-robust	nieokreślone zach.	nieokreślone zach.
DEFAULT	robust	nieokreślone zach.	zwrócenie błędu

Tworzenie/usuwanie zmiennych warunkowych

- int pthread_cond_init(pthread_cond_t *restrict cond, const pthread_condattr_t *restrict attr)
  - Inicjalizuje zmienną wskazaną przez cond do wykorzystania w charakterze zmiennej warunkowej
  - Opcjonalnie można podać atrybuty zmiennej warunkowej w zmiennej wskazanej przez attr (jeśli chcemy pozostawić wartości domyślne, możemy jako argument attr podać NULL)
- pthread_cond_t cond = PTHREAD_COND_INITIALIZER;
  - Makro tworzące zmienną warunkową z domyślnymi atrybutami
- int pthread_cond_destroy(pthread_cond_t *cond)
  - Sprząta zasoby związane ze wskazaną zmienną warunkową cond
- Wszystkie powyższe konstrukcje opisane są w man 3 pthread_cond_destroy

Atrybuty zmiennej warunkowej

- Zmienna typu pthread_condattr_t
- int pthread_condattr_init(pthread_condattr_t *attr)
  - Inicjalizuje wskazaną zmienną attr do wykorzystania jako atrybuty zmiennej warunkowej
- int pthread_condattr_destroy(pthread_condattr_t *attr)
  - Sprząta zasoby które zaalokowano w związku z zainicjalizowaniem zmiennej wskazanej przez attr do wykorzystania jako atrybuty zmiennej warunkowej
- Obie powyższe funkcje opisane w man 3 pthread_condattr_destroy

Atr. zm. warunkowej - dzielenie przez pamięć wsp.

- int pthread_condattr_getpshared(const pthread_condattr_t *restrict attr, int *restrict pshared)
- int pthread_condattr_setpshared(pthread_condattr_t *attr, int pshared)
- Stałe dla argumentu pshared:
  - PTHREAD_PROCESS_SHARED - zmienna warunkowa może być wykorzystywana przez wszystkie wątki ze wszystkich procesów mających dostęp do obszaru pamięci wspólnej w którym umieszczono zmienną warunkową
  - **PTHREAD_PROCESS_PRIVATE** - zmienna warunkowa może być wykorzystywana wyłącznie w ramach wątków należących do pojedynczego procesu (domyślne)
- Obie funkcje opisane w man 3 pthread_condattr_getpshared

Oczekiwanie na zmienną warunkową

- int pthread_cond_wait(pthread_cond_t *restrict cond, pthread_mutex_t *restrict mutex)
  - Zasypia w oczekiwaniu na zasygnalizowanie zmiennej warunkowej cond i zwalnia mutex na czas oczekiwania
- int pthread_cond_timedwait(pthread_cond_t *restrict cond, pthread_mutex_t *restrict mutex, const struct timespec *restrict abstime)
  - Zasypia w oczekiwaniu na zasygnalizowanie zmiennej warunkowej cond i zwalnia mutex na czas oczekiwania; jeśli warunek nie zostanie zasygnalizowany w ciągu abstime, zostaje zwrócony błąd ETIMEDOUT
- Obie powyższe konstrukcje opisane są w man 3 pthread_cond_timedwait
- W kodzie programu stosowane powinny być wyłącznie w formie:

  1) zajmij mutex 2) sprawdź zmienną skojarzoną ze zmienną warunkową
  3*) [oczekuj na zasygnalizowanie warunku, ponownie sprawdź] 4) zwolnij mutex

Sygnalizacja zmiennej warunkowej

- int pthread_cond_signal(pthread_cond_t *cond)
  - Sygnalizuje warunek jednemu z wątków które na niego oczekują
- int pthread_cond_broadcast(pthread_cond_t *cond)
  - Sygnalizuje warunek wszystkim wątkom które na niego oczekują
- Obie powyższe funkcje opisane są w man 3 pthread_cond_broadcast
- W kodzie programu podobnie jak pthread_cond_wait, powinny być otoczone zajęciem i zwolnieniem muteksu:

  1) zajmij mutex
  2) zmodyfikuj zmienną skojarzoną ze zmienną warunkową
  3) sygnalizuj/rozgłoś warunek
  4) zwolnij mutex

Dziękuję za uwagę