Mutex

Тип: pthread_mutex_t

Жизнь:

- **инициализация**:
  pthread_mutex_init или
  PTHREAD_MUTEX_INITIALIZER
- **запрос на монопольное использование**:
  pthread_mutex_lock или
  pthread_mutex_timedlock
- **отказ от монопольного использования**:
  pthread_mutex_unlock
- **тестирование монопольного использования**:
  pthread_mutex_trylock
- **разрушение**:
  pthread_mutex_destroy

Атрибуты

Тип: pthread_mutexattr_t

Жизнь:

- **инициализация**: pthread_mutexattr_init
- **запрос атрибута**: pthread_mutexattr_get...
- **установка атрибута**:
- pthread_mutexattr_set...
- **разрушение**: pthread_mutexattr_destroy

Примеры атрибутов:

- **принадлежность**: PTHREAD_PROCESS_SHARED
- **рекурсивность**: PTHREAD_MUTEX_RECURSIVE

Чтение/запись

Тип: pthread_rwlock_t

Жизнь:

- **инициализация**:
  pthread_rwlock_init
- **запрос на чтение**:
  pthread_rwlock_rdlock, pthread_rwlock_timedrdlock
- **запрос на запись**:
  pthread_rwlock_wrlock, pthread_rwlock_timedwrlock
- **освобождение**:
  pthread_rwlock_unlock
- **тестирование**:
  pthread_rwlock_tryrdlock, pthread_rwlock_trywrlock
- **разрушение**:
  pthread_rwlock_destroy

Условные переменные

Тип: pthread_cond_t

Жизнь:

- **инициализация**:
  pthread_cond_init или
  PTHREAD_C OND_INITIALIZER
- **ожидание**
  pthread_cond_wait или
  pthread_cond_timedwait
- **сигнализация**:
  pthread_cond_signal или pthread_cond_broadcast
- **разрушение**:
  pthread_cond_destroy

```
#define SYNC_MAX_COUNT  10

void SynchronizationPoint() {
        static mutex_t sync_lock = PTHREAD_MUTEX_INITIALIZER;
        static cond_t  sync_cond = PTHREAD_COND_INITIALIZER;
        static int sync_count = 0;

        /* блокировка доступа к счетчику */
        pthread_mutex_lock(&sync_lock);

        sync_count++;

        /* проверка: следует ли продолжать ожидание */
        if (sync_count < SYNC_MAX_COUNT)
                pthread_cond_wait(&sync_cond, &sync_lock);
        else
                /* оповестить о достижении данной точки всеми */
                pthread_cond_broadcast(&sync_cond);

        /* активизация взаимной блокировки - в противном случае
                из процедуры сможет выйти только одна нить! */
        pthread_mutex_unlock(&sync_lock);
}
```