

# Механизмы синхронизации потоков. Мьютексты

```
int pthread_mutex_init
```

```
(pthread_mutex_t *mutex, const pthread_mutexattr_t *mutexattr)
```



(OUT) Идентификатор мьютекса

- Схема бинарных значений (0/1)
- В каждый момент времени мьютексом владеет только один поток
- Поток, захвативший мьютекс, после прохождения критической секции обязан его отпустить
- Все операции с мьютексом атомарны

# Механизмы синхронизации потоков. Мьютексты

```
int pthread_mutex_init (pthread_mutex_t *, const pthread_mutexattr_t *)
```

```
int pthread_mutex_lock (pthread_mutex_t *)
```

```
int pthread_mutex_unlock (pthread_mutex_t *)
```

```
int pthread_mutex_destroy (pthread_mutex_t *)
```

# Механизмы синхронизации потоков. Семафоры

```
#include <semaphore.h>
```

```
int sem_init (sem_t *sem, int pshared, unsigned int value)
```



(OUT) Идентификатор семафора



- 0 – локален для процесса
- 1 – общий между процессами



Начальное значение

- Схема целых положительных чисел
- Не имеет потока-владельца
- Все операции с семафором атомарны

# Механизмы синхронизации потоков. Семафоры

```
#include <semaphore.h>
```

```
int sem_init (sem_t *, int, unsigned int)
```

```
int sem_destroy (sem_t *)
```

```
int sem_post (sem_t *)
```

```
int sem_wait (sem_t *)
```

```
int sem_getvalue (sem_t *, int*)
```

## Общие процедуры PThread. Задача 2

- Составить программу, суммирующую все натуральные числа от 1 до N
- Каждый поток получает свой диапазон чисел для суммирования
- N задается аргументом запуска
- Вести подсчет времени выполнения всей программы + времени подсчета на каждом потоке

## Общие процедуры PThread. Задача 3

- Составить программу, выполняющую интегрирование по частям функции  $x dx$
- Вести подсчет времени выполнения всей программы + времени подсчета на каждом потоке
- Шаг и размер частей задается при запуске
- Вычисление более сложного интеграла + итоговый результат готов после выполнения последнего потока (master thread не производит никаких вычислений)

# Время работы программы

```
#include<time.h> // заголовочный файл, содержащий типы и функции  
для работы с датой и временем
```

```
struct timespec begin, end; // требует использования ключа -lrt !  
double elapsed;
```

```
clock_gettime(CLOCK_REALTIME, &begin); /* возвращает ссылку на  
запись типа timespec, которая  
объявлена в time.h с полями  
time_t tv_sec; – секунды,  
long tv_nsec; – наносекунды.  
*/
```

```
... // здесь работают нити
```

```
clock_gettime(CLOCK_REALTIME, &end);
```

```
elapsed = end.tv_sec - begin.tv_sec; // время в секундах  
elapsed += (end.tv_nsec - begin.tv_nsec) / 1000000000.0; // добавляем  
время вплоть до наносекунд
```