Средства и системы Параллельного программирования

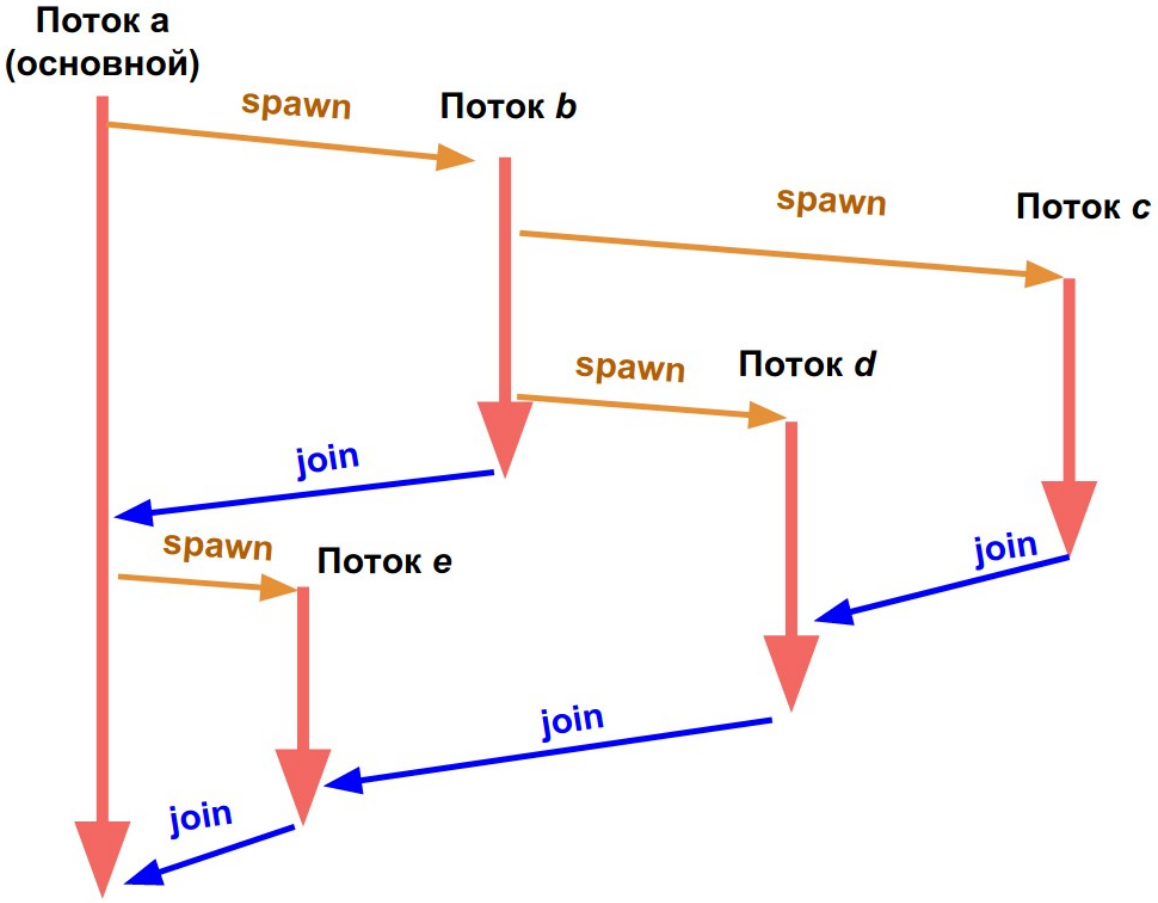
Хамитов Камиль
berserq0123@gmail.com

Библиотека pthreads

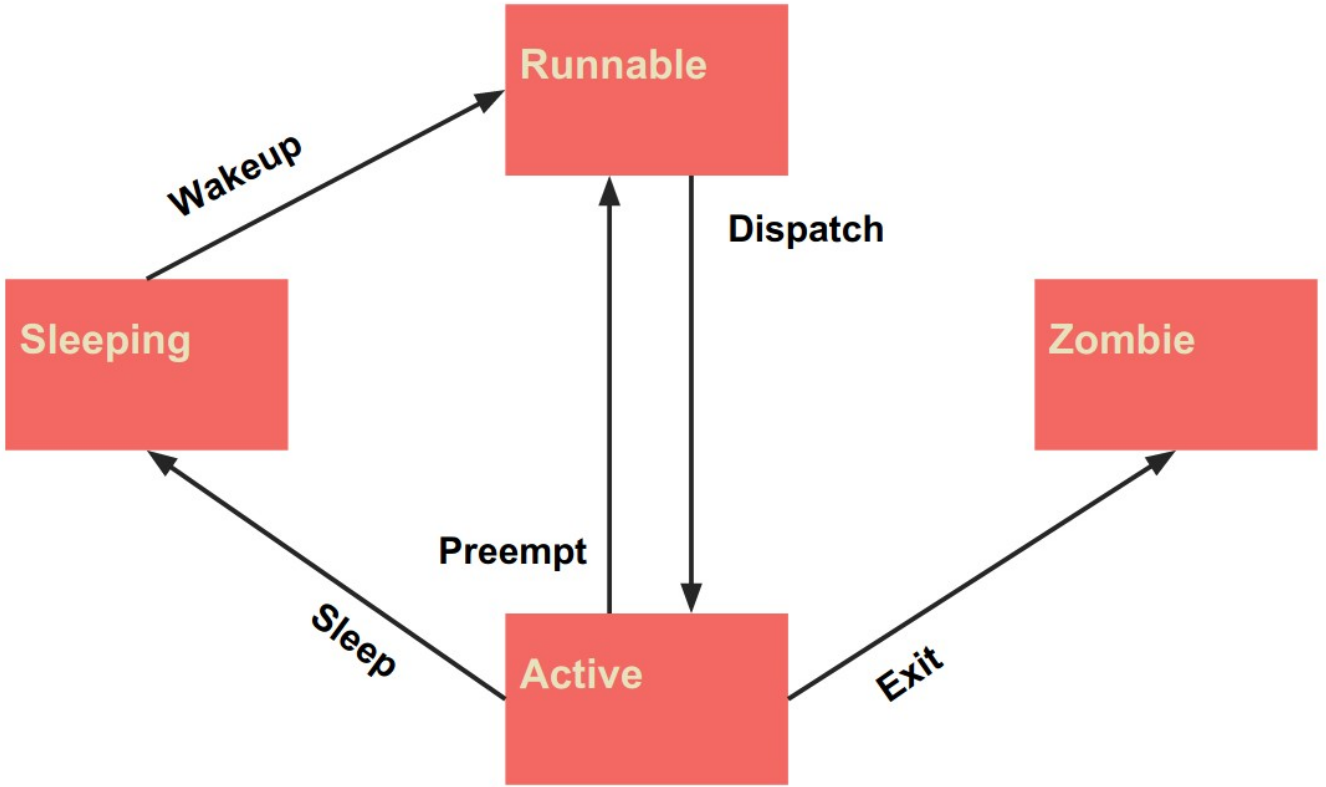
- Низкоуровненвая библиотека работы с потоками – (поток последовательность комманд, которая может быть запланирована ОС независимо)
- Подключить заголовочный файл <pthread.h>
- Функции pthread_*
- Собирать с помощью cc -Wall -pedantic -O2 -pthread \

  -o prog prog.c

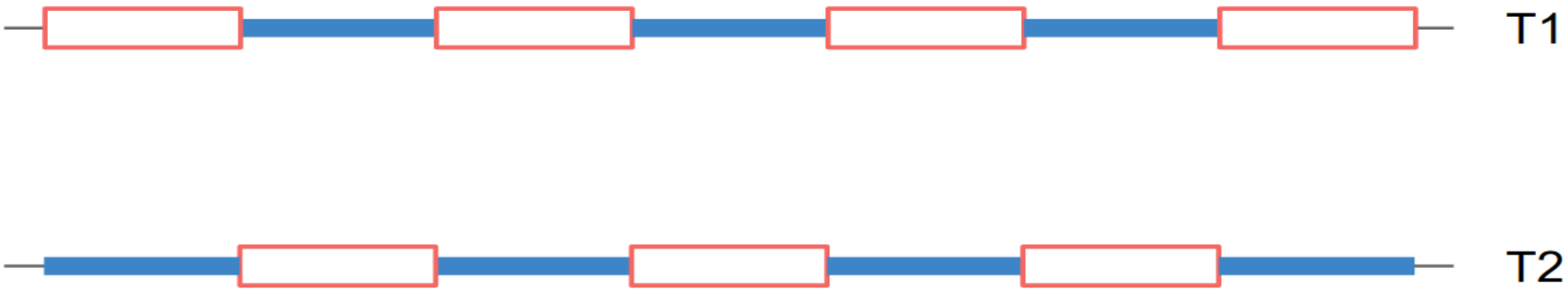
Жизненный цикл потока



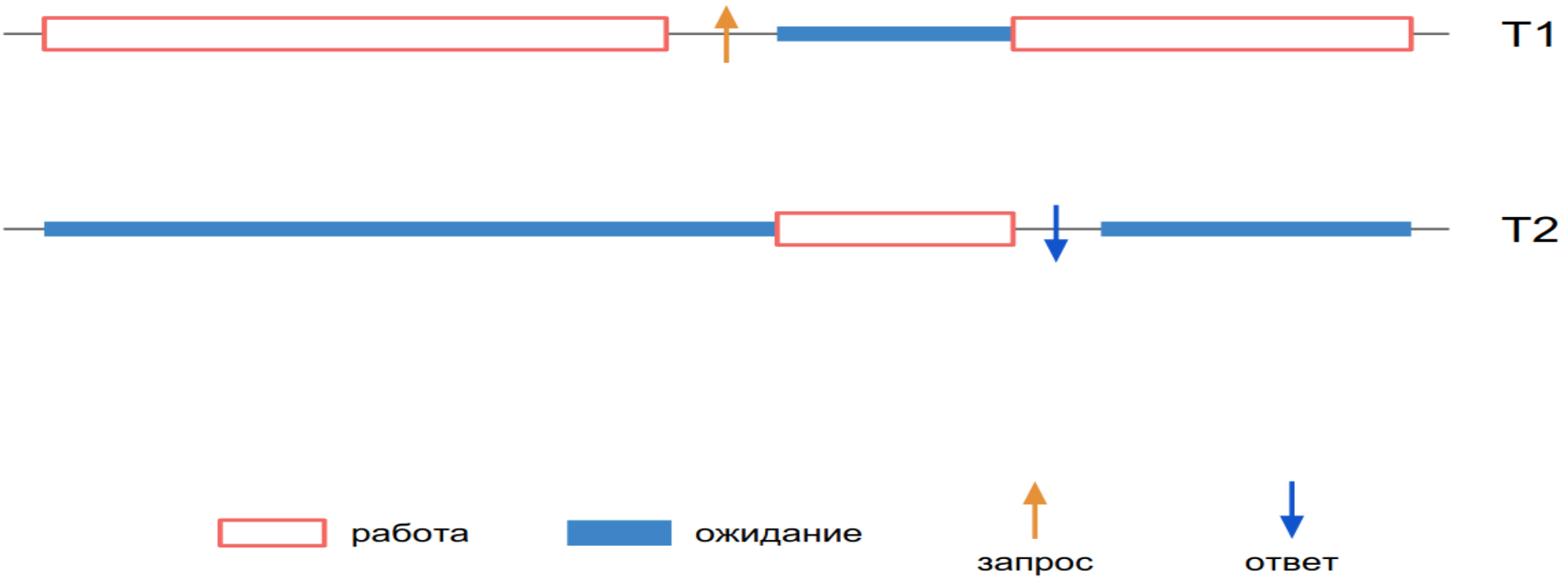
Жизненный цикл потока



Планирование потоков



Планирование с приоритетами



Активации потока планировщиком

- Синхронизация – T1 запрашивает мьютекс, и если он занят потоком T2, то T1 встаёт в очередь, тем самым давая возможность другим потокам запуститься
- Вытеснение – Происходит событие, в результате которого высокоприоритетный поток T2 может запуститься. Тогда поток T1 с низким приоритетом вытесняется, и T2 запускается
- Уступание – явно вызван sched_yield() во время работы T1, и тогда планировщик ищет другой поток T2, который может запуститься, и запускает его
- Квантование – Квант времени для потока T1 истёк. Тогда поток T2 получает квант и запускается.

Функции планирования

```
#include <pthread.h>

int pthread_setschedparam(pthread_t thread, int policy,
                          const struct sched_param *param);
int pthread_getschedparam(pthread_t thread, int *restrict policy,
                          struct sched_param *restrict param);
```

```
struct sched_param {
    int sched_priority;     /* Scheduling priority */
};
```

0 – Normal(SCHED_OTHER/ SCHED_BATCH/SCHED_IDLE)
1..99 – realtime(SCHED_FIFO/SCHED_RR/SCHED_DEADLINE

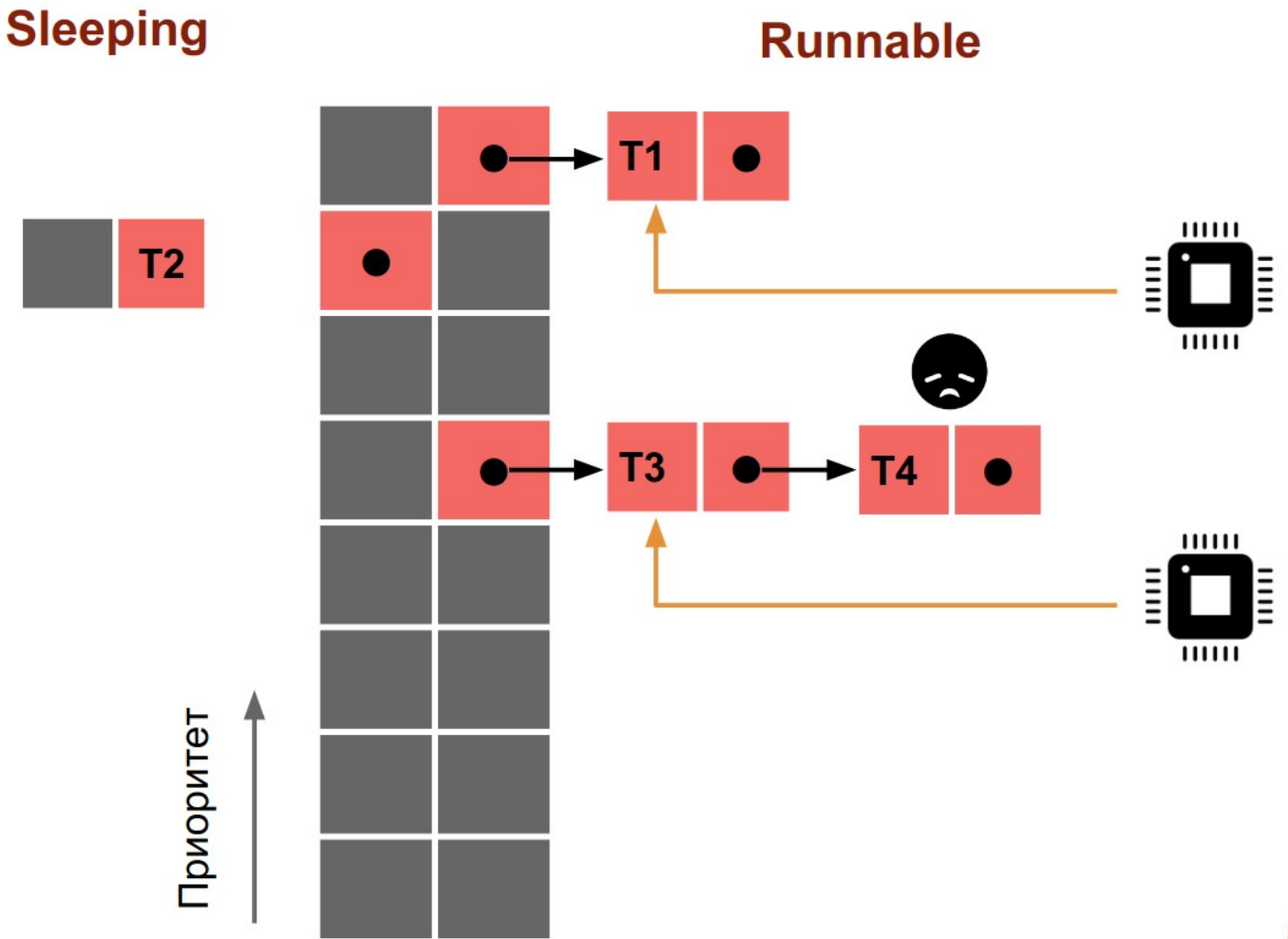
Планирование normal

- SCHED_OTHER – политика по-умолчанию, динамический приоритет, использует nice значение от setpriority/sched_setattr), fair по планированию
- SCHED_BATCH – через систему штрафов за активацию потока, предполагается что все потоки cpu-intensive. Для неинтерактивных задач с большим объемом вычислений или пакетной обработки данных
- SCHED_IDLE – низкоприоритетные задачи

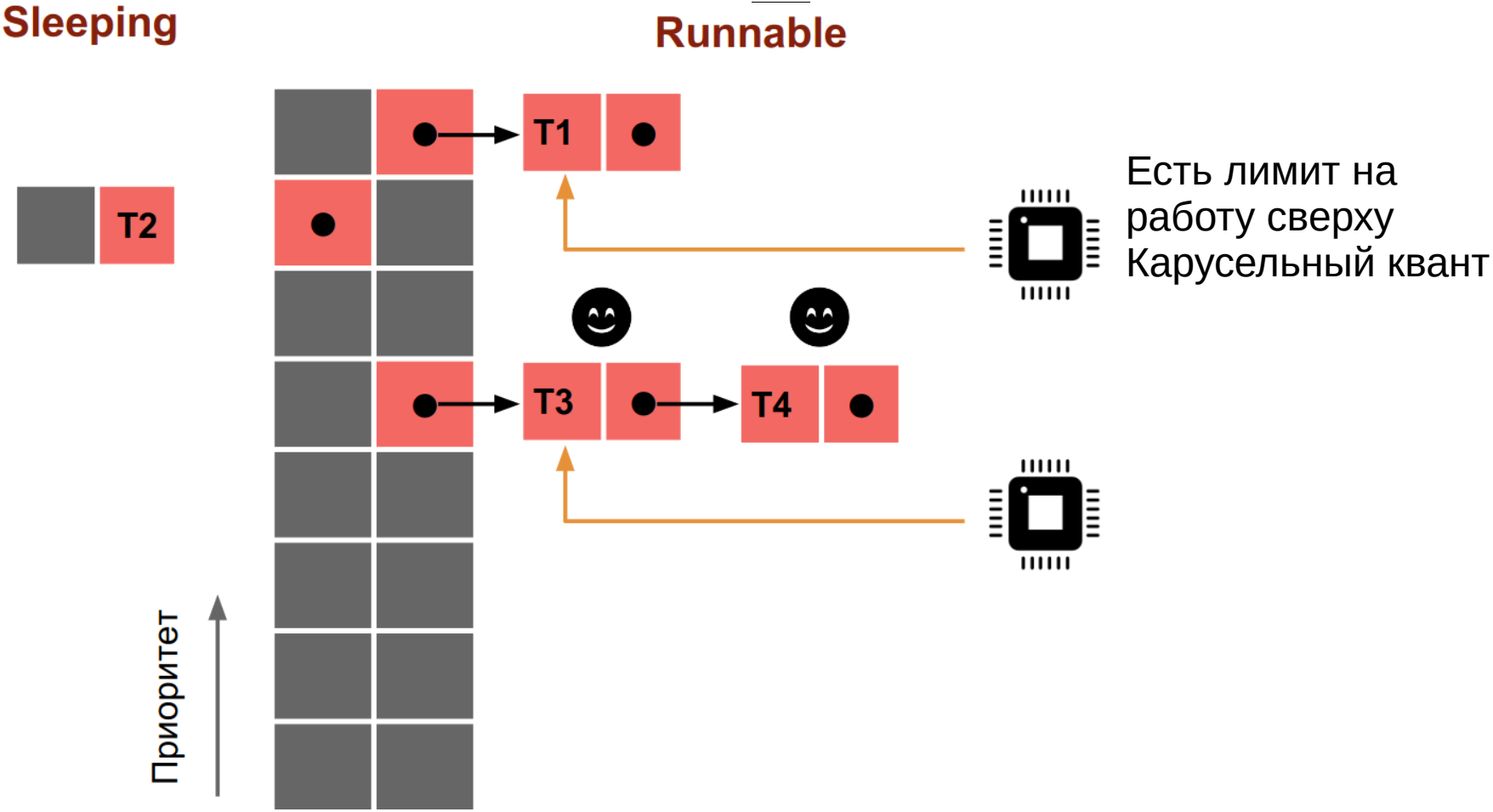
Планирование normal

- SCHED_OTHER – политика по-умолчанию, динамический приоритет, использует nice значение от setpriority/sched_setattr), fair по планированию
- SCHED_BATCH – через систему штрафов за активацию потока, предполагается что все потоки cpu-intensive. Для неинтерактивных задач с большим объемом вычислений или пакетной обработки данных
- SCHED_IDLE – низкоприоритетные задачи

SCHED FIFO



SCHED_RR

- 

Есть лимит на работу сверху
Карусельный квант

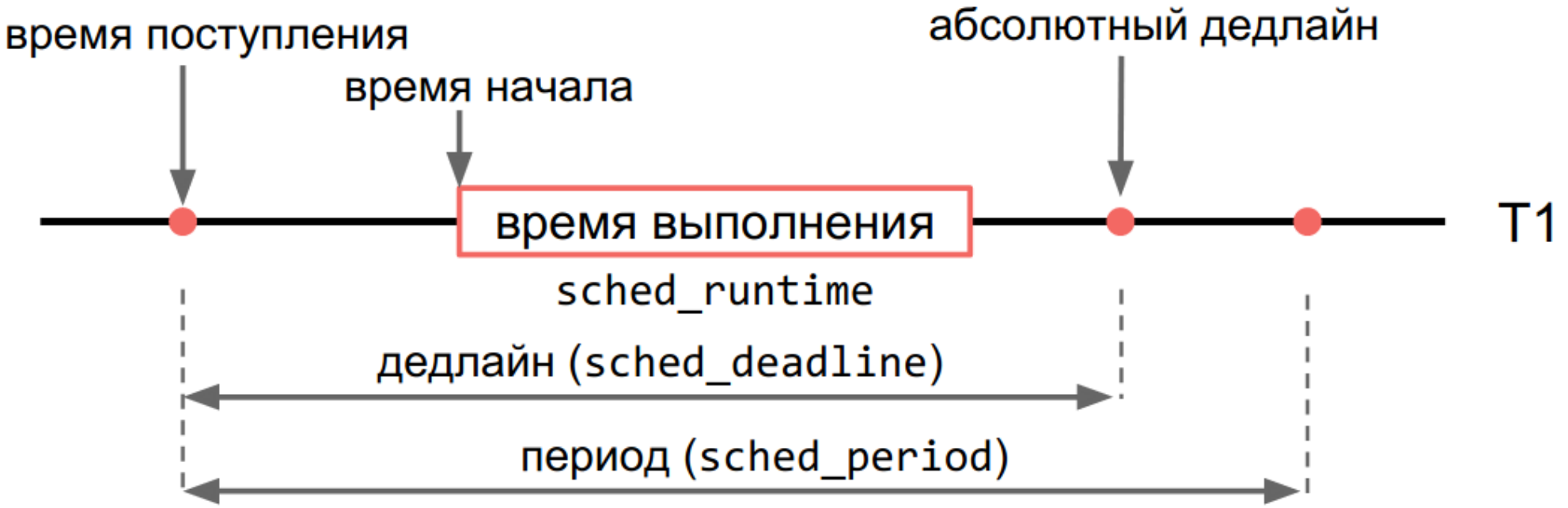
SCHED_DEADLINE

- Модель использующая свойства спорадических групп.
- 1 задача запускается не более 1 раза за период
- Каждая задача имеет относительный deadline сверху(когда должна завершится) и максимальное время вычислений что она проведет на CPU

SCHED_DEADLINE

- Модель использующая свойства спорадических групп.
- 1 задача запускается не более 1 раза за период
- Каждая задача имеет относительный deadline сверху(когда должна завершится) и максимальное время вычислений что она проведет на CPU

SCHED_DEADLINE



Блокировки

Поток 1

```
bal = GetBalance(account);
bal += bal * InterestRate;

PutBalance(account, bal);
```

Поток 2

```
bal = GetBalance(account);
bal += deposit;
PutBalance(account, bal);
```

Критическая секция – участок кода который должен быть выполнен полностью без прерываний

Все общедоступные для модификации несколькими потоками данные **должны** быть защищены

Функции работы с mutex

```
int pthread_mutex_lock(pthread_mutex_t *mutex);
int pthread_mutex_trylock(pthread_mutex_t *mutex);
int pthread_mutex_unlock(pthread_mutex_t *mutex);
int pthread_mutex_destroy(pthread_mutex_t *mutex);
int pthread_mutex_init(pthread_mutex_t *restrict mutex,
    const pthread_mutexattr_t *restrict attr);
pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;
```

Livecoding demo session