

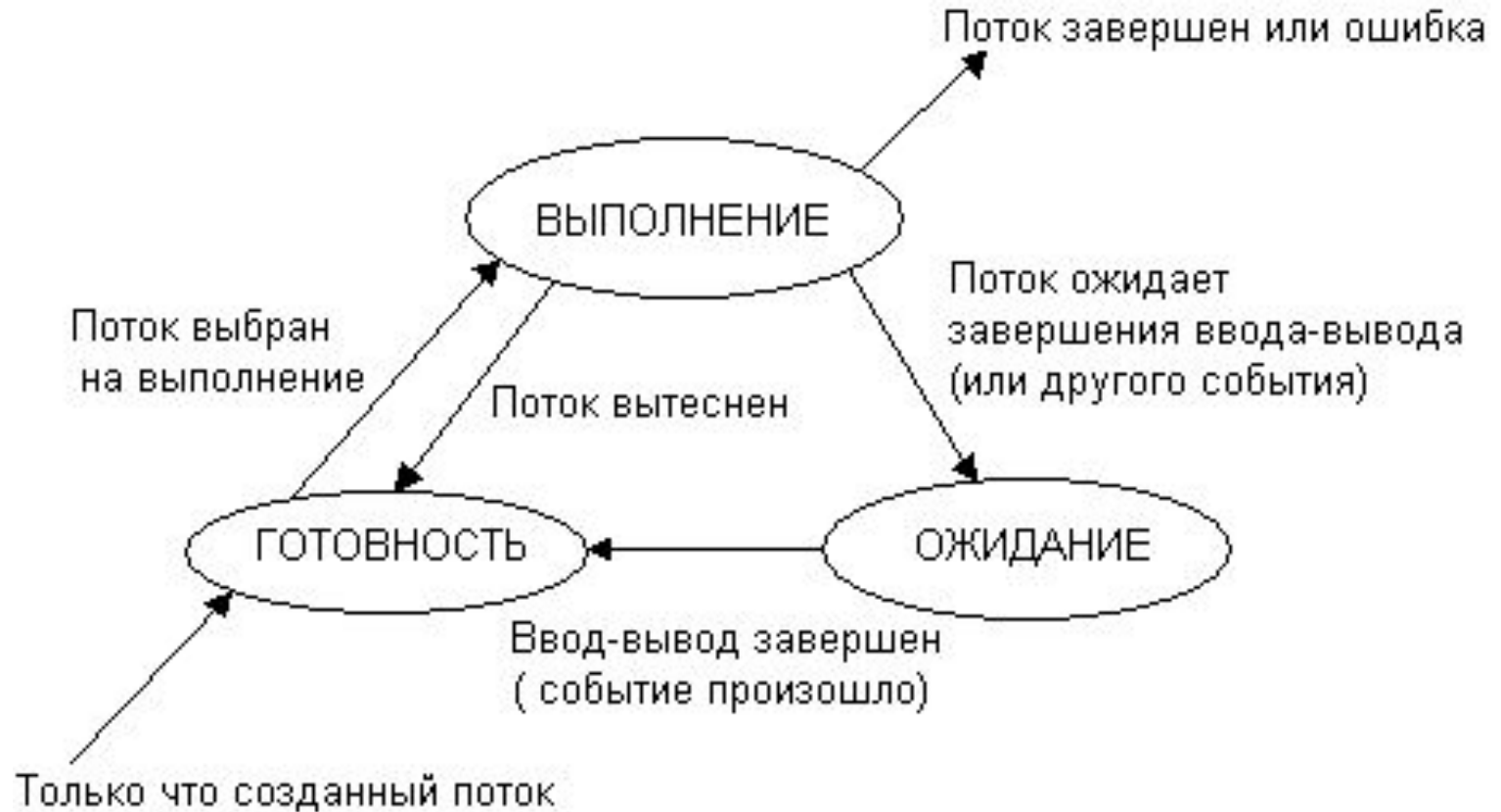
Операционные системы реального времени

Управление процессами и
потоками

Процессы и потоки в ОС РВ

- Под понятием *задачи* в терминах ОС и программных комплексов могут пониматься два типа единиц работы: *процессы* и *потоки (нити)*.
- *Процесс* - обобщенное представление задачи, независимый модуль программы или весь исполняемый файл целиком с его адресным пространством, состоянием регистров процессора, счетчиком команд, кодом процедур и функций.
- *Поток* - составная часть процесса, обозначает последовательность исполняемого кода.

Состояния потока



Типичный граф состояний потока в многозадачной среде

Диспетчеризация потоков

Потоки имеют одинаковый приоритет:

1. *FIFO (First Input First Output) – Первый Вошел Первый Вышел.*
2. *Карусельная многозадачность (round robin).*

Потоки имеют разный приоритет:

3. *Приоритетная многозадачность,*
4. *Адаптивная многозадачность.*
5. *Вытесняющая приоритетная многозадачность.*

*ОСРВ должна обеспечивать многозадачность с поддержкой **вытесняющей приоритетной методики диспетчеризации***

ОС должна иметь достаточно большое (определяется масштабом задачи) количество приоритетов

Читаем примечания:

Механизмы синхронизации

Взаимное исключение

Самые общие методы получения исключительного доступа к разделяемым ресурсам - это:

- запрещение прерываний;
- проба - и - установка;
- отключение диспетчеризации;
- использование семафоров.

Запрещение и разрешение прерываний

1. Запретить прерывания;
2. Осуществить доступ (чтение/запись переменных);
3. Разрешить прерывания;

Проба - и - Установка

```
Запретить прерывания;
    Если ('Переменная доступа'== 0)
{
    Установить переменную в 1;
    Разрешить прерывания;
    /*Доступ разрешён*/
    Произвести доступ к ресурсам;
    Запретить прерывания;
    Установить переменную доступа в ноль;
    Разрешить прерывания;
} иначе
{
    Разрешить прерывания;
    /*Доступ запрещён, попробуйте позже*/
}
```

Блокировка диспетчеризации

```
void Function (void)
{ OSSchedLock(); . /*Здесь доступ к разделяемым данным
    разрешён, прерывания разрешены */ .
OSSchedUnlock();
}
```

Семафоры

Семафоры применяются чтобы:

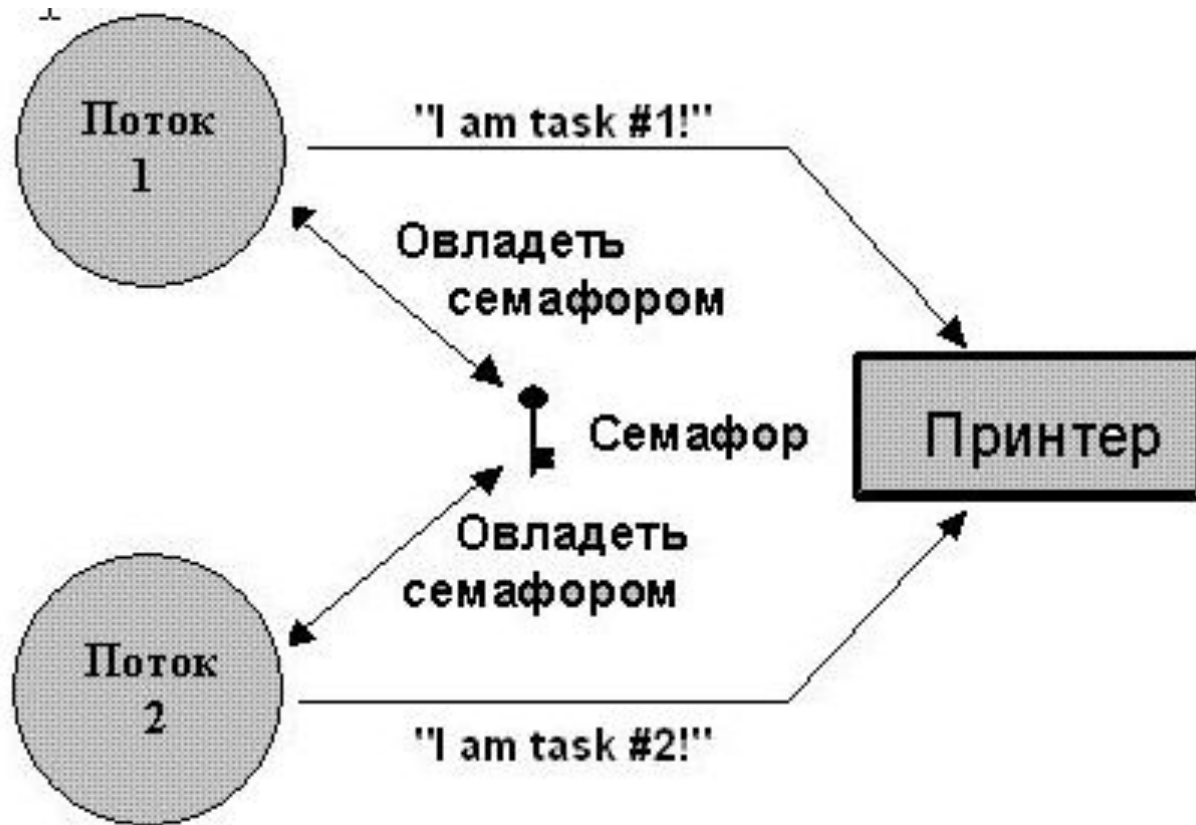
- управлять доступом к разделяемым ресурсам;
- сигнализировать наступление события;
- позволять двум потокам синхронизировать их деятельность.

Семафор - это ключ, которым должен овладеть поток, чтобы продолжить выполнение.

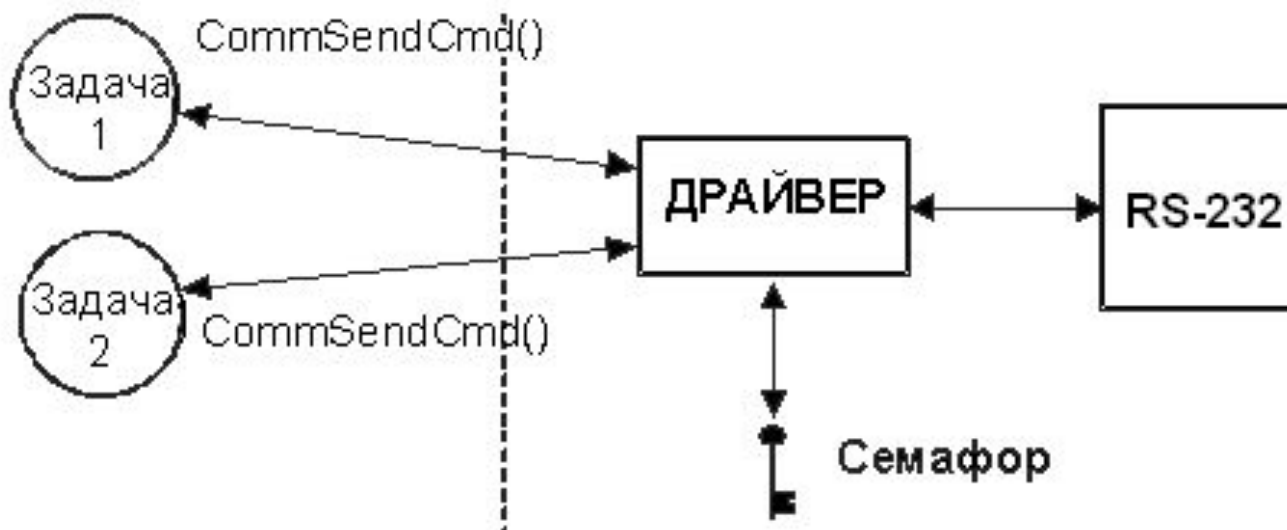
Есть два типа семафоров: *двоичные* и *счётные*.

Семафоры

Существуют три основные операции, которые можно производить с семафорами - это Инициализация *INITIALIZE* (или *CREATE*), Ожидание *WAIT* (или *PEND*) и Освобождение *SIGNAL* (или *POST*).

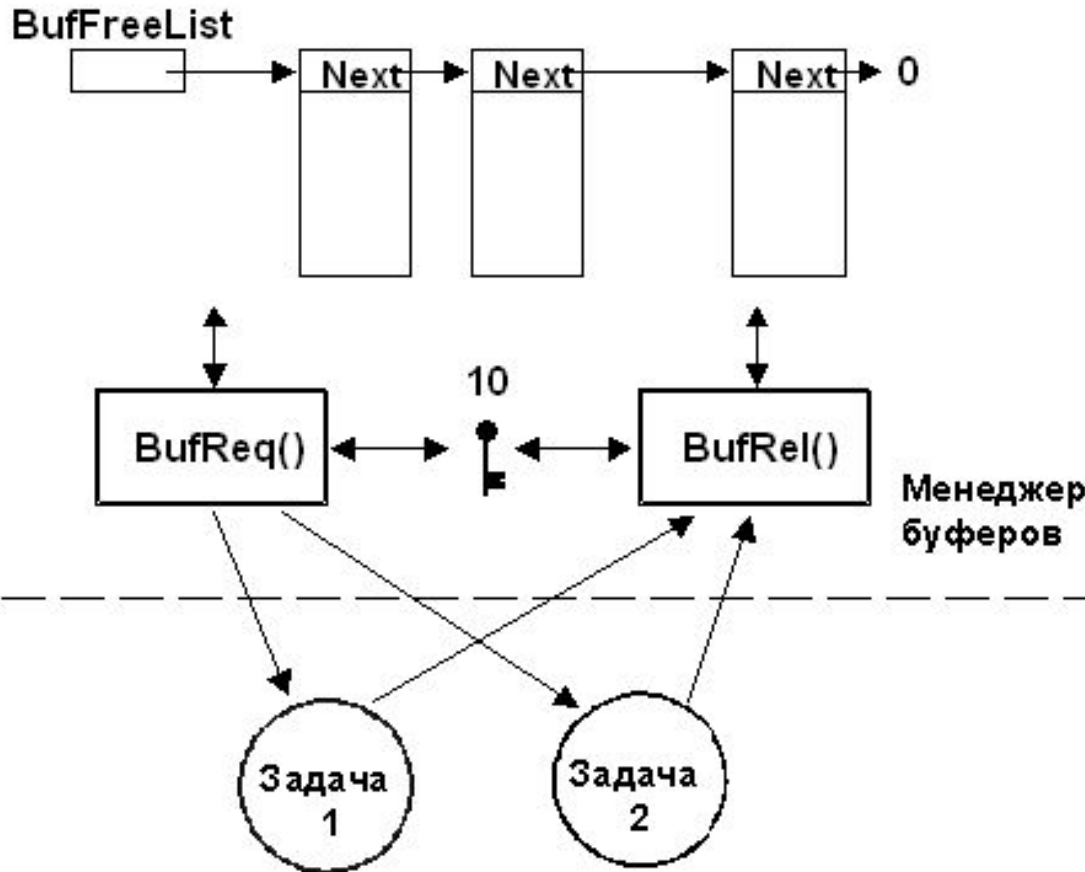


Скрытый семафор



```
INT8U CommSendCmd(char *cmd, char *response, INT16U timeout)
{
    Овладеть семафором порта;
    Послать команду устройству;
    Ожидать ответа(timeout);
    если(таймаут вышел) { Освободить семафор; Возвратить
        код ошибки; }
    иначе { Освободить семафор;
        Возвратить код отсутствия ошибки; } }
```

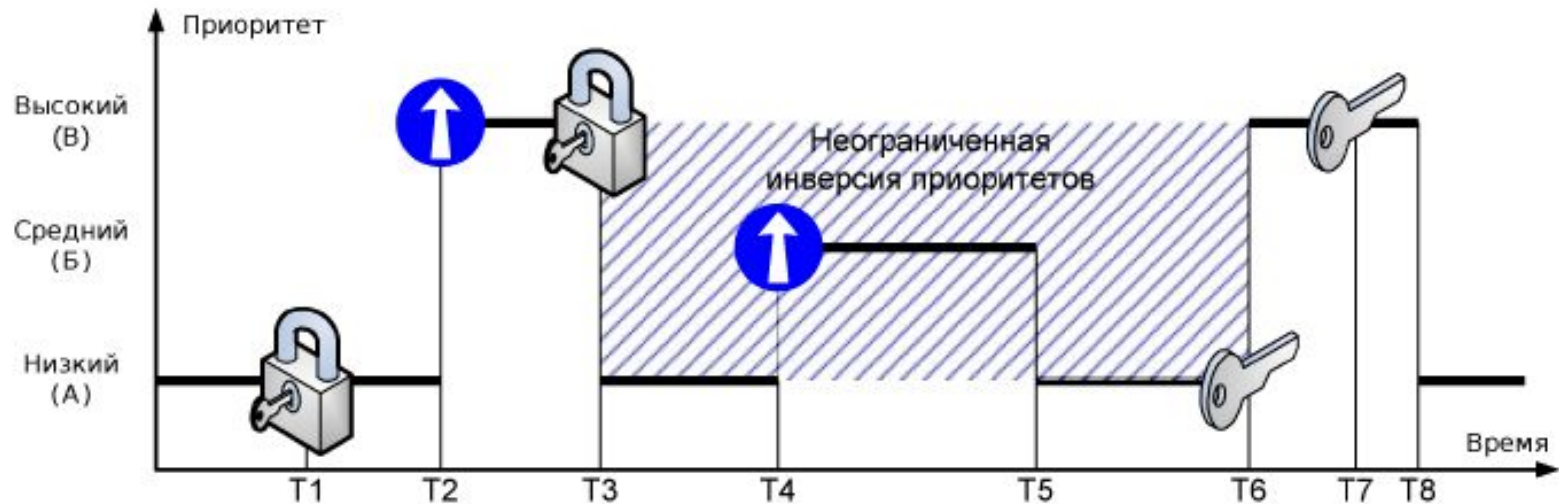
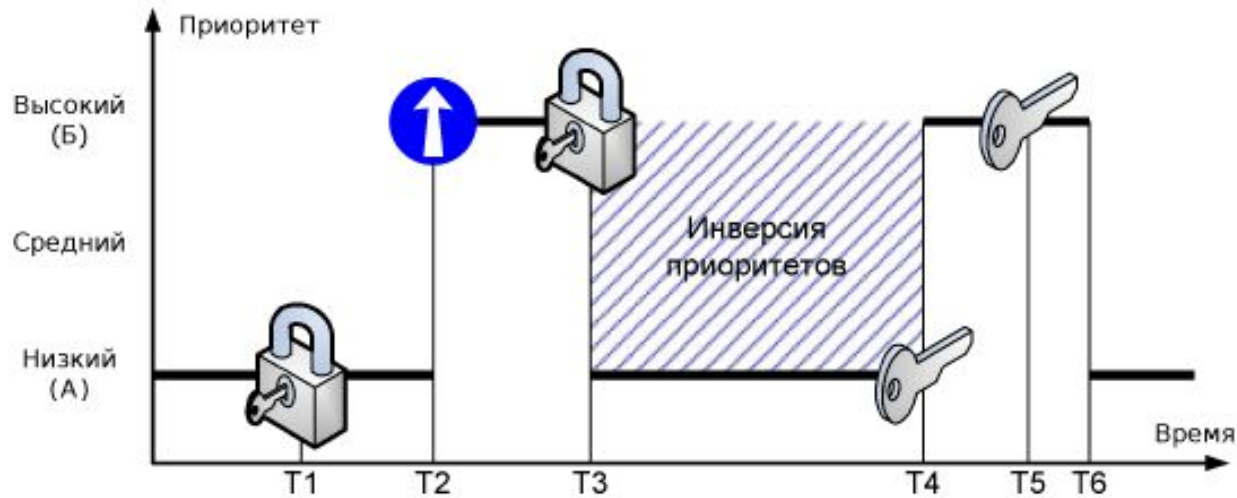
Применение счётного семафора



```
BUF *BufReq(void)
{
    BUF *ptr;
    Овладеть семафором;
    Запретить прерывания;
    ptr      = BufFreeList;
    BufFreeList  = ptr->BufNext;
    Разрешить прерывания;
    Возвратить (ptr);
}

void BufRel(BUF *ptr)
{
    Запретить прерывания;
    ptr->BufNext = BufFreeList;
    BufFreeList = ptr;
    Разрешить прерывания;
    Освободить семафор;
}
```

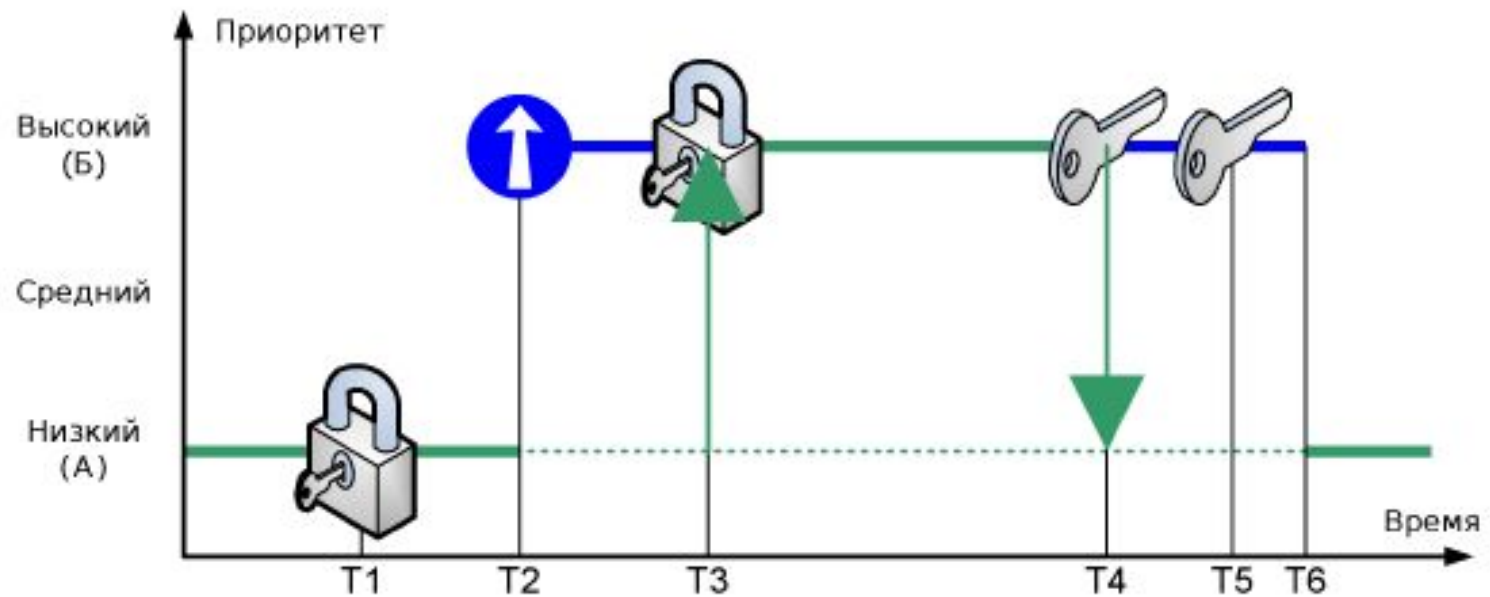
Инверсия приоритетов



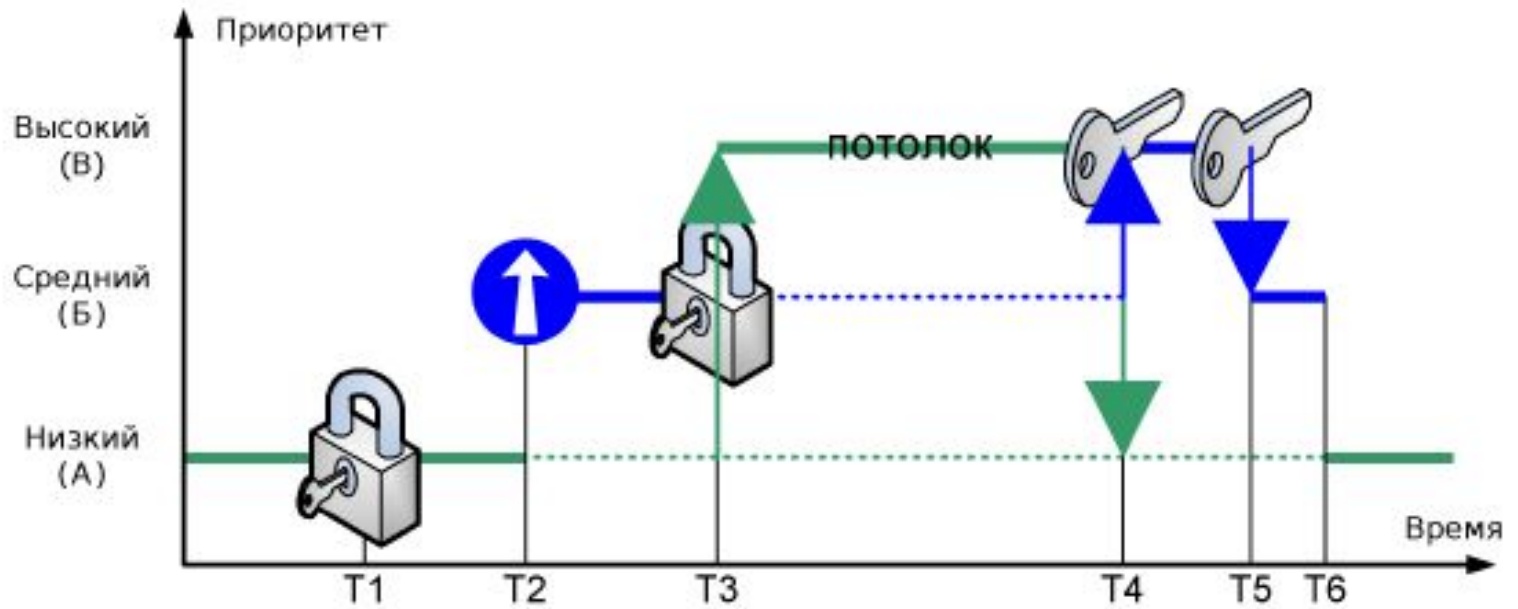
Защита от инверсии приоритетов

- *Наследование приоритетов - наследование низкоприоритетным потоком, захватившим ресурс, приоритета от высокоприоритетного потока, которому этот ресурс нужен*
- *Протокол Предельного Приоритета (Priority Ceiling Protocol) - добавление к стандартным свойствам объектов синхронизации параметра, определяемого максимальным приоритетом потока, которые к этому объекту обращаются*

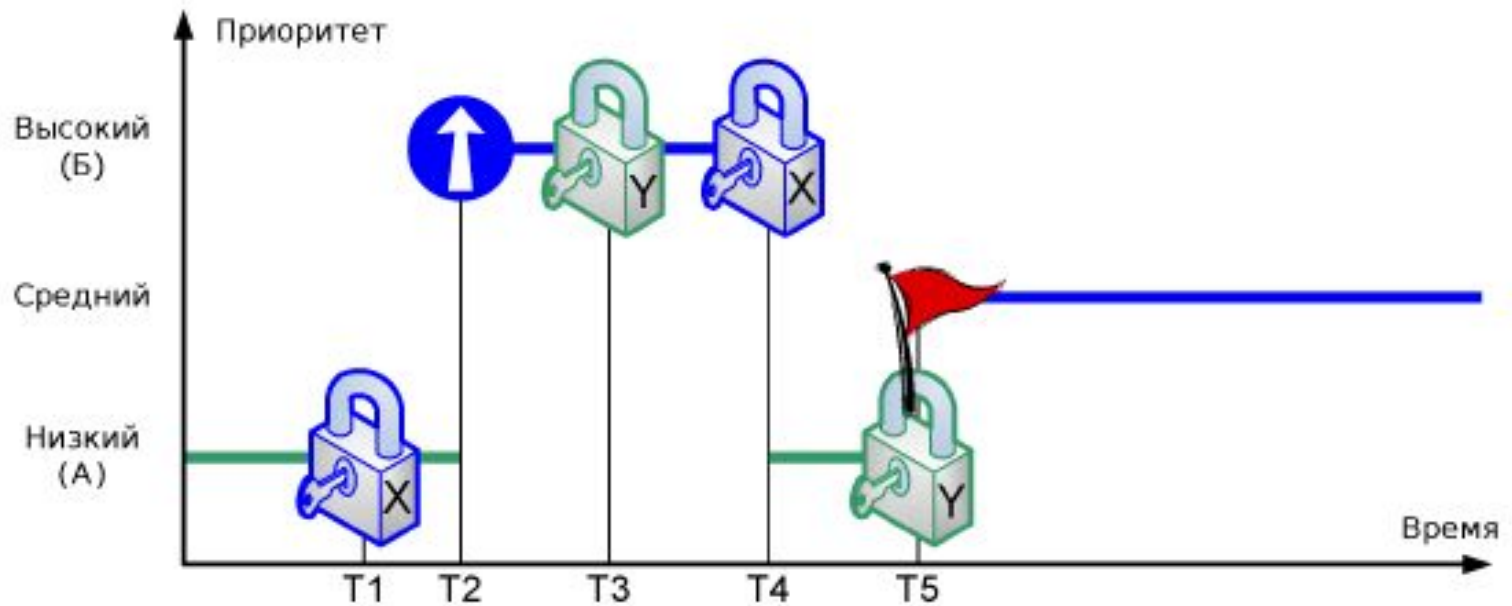
Протокол наследования приоритета



Протокол увеличения приоритета



Взаимная блокировка



Временные характеристики ОСРВ

- время отклика на прерывание — время между фактическим возникновением прерывания и началом обработки первой инструкции обработчика прерывания;
- время переключения потока управления — время переключения между двумя потоками в одном процессе;
- время переключения контекста процесса (только для ОС, поддерживающих модель процессов) — время переключения между двумя потоками управления, принадлежащими двум различным процессам.

ОСРВ должна обеспечивать стабильность временных параметров.

| Время реакции | Варианты ОС |
|----------------------|---|
| Менее 10 мкс | Только ОСРВ - это граница выбора между схемным и программным решениями |
| 10 – 100 мкс | Операционные системы реального времени |
| 100мкс – 1 мс | ОСРВ, RTAI, RT LINUX, расширения реального времени для Windows NT, CE |
| Более 1 мс | Linux, Windows NT, только для задач "мягкого" реального времени, где опоздания реакции не могут привести к тяжелым последствиям |

Современные ОСРВ

- **ОСРВ VxWorks AE 1.1.** Построена по принципам **МОНОЛИТНОЙ** операционной системы. Поддерживает **приоритетную** вытесняющую многозадачность в комбинации с **карусельной** многозадачностью. Новшество - *защищенные домены*.
- **ОСРВ Windows CE.NET.** Архитектура соответствует **МОНОЛИТНОЙ** модели архитектуры ОС, однако для повышения **масштабируемости** часть сервисов системы оформлены как отдельные модули, взаимодействующие с ядром по технологии COM. Система поддерживает вытесняющую **приоритетную** многозадачность в комбинации с **карусельной** и **FIFO** многозадачностью.

- **QNX 6.5** Строится на базе микроядра с организованными по технологии клиент – сервер сервисами, вынесенными на уровень пользовательских приложений. Микроядро системы выступает в качестве диспетчера сообщений, переадресовывая системные вызовы прикладных программ клиентов к соответствующим серверам сервисов и обратно.

Результаты сравнительного тестирования ОСРВ QNX 6.1, VxWorks и Windows CE.NET

| Тест | QNX 6.1 (мкс) | | VxWorks AE 1.1 (мкс) | | Windows CE.NET (мкс) | |
|---|---------------|----------|----------------------|----------|----------------------|----------|
| | Среднее | Максимум | Среднее | Максимум | Среднее | Максимум |
| Время переключения контекста для 2 потоков в одном процессе | 2,0 | 8,1 | 2,9 | 15,5 | 2,6 | 35,1 |
| Время переключения контекста для 10 потоков в одном процессе | 2,4 | 7,4 | 3,4 | 16,5 | 3,3 | 37,4 |
| Время переключения контекста для 128 потоков в одном процессе | 3,3 | 11,6 | 6,5 | 29,6 | 5,3 | 63,9 |
| Время переключения контекста для 128 потоков в разных процессах | 7,2 | 15,9 | 6,8 | 46,8 | 9,6 | 16,7 |