Учебный курс

Операционные системы

Тема 2. Процессы и потоки. Планирование и синхронизация

2.1. Концепция процессов и потоков. Задания, процессы, потоки (нити), волокна

2.2. Мультипрограммирование. Формы многопрограммной работы

2.3. Управление процессами и потоками

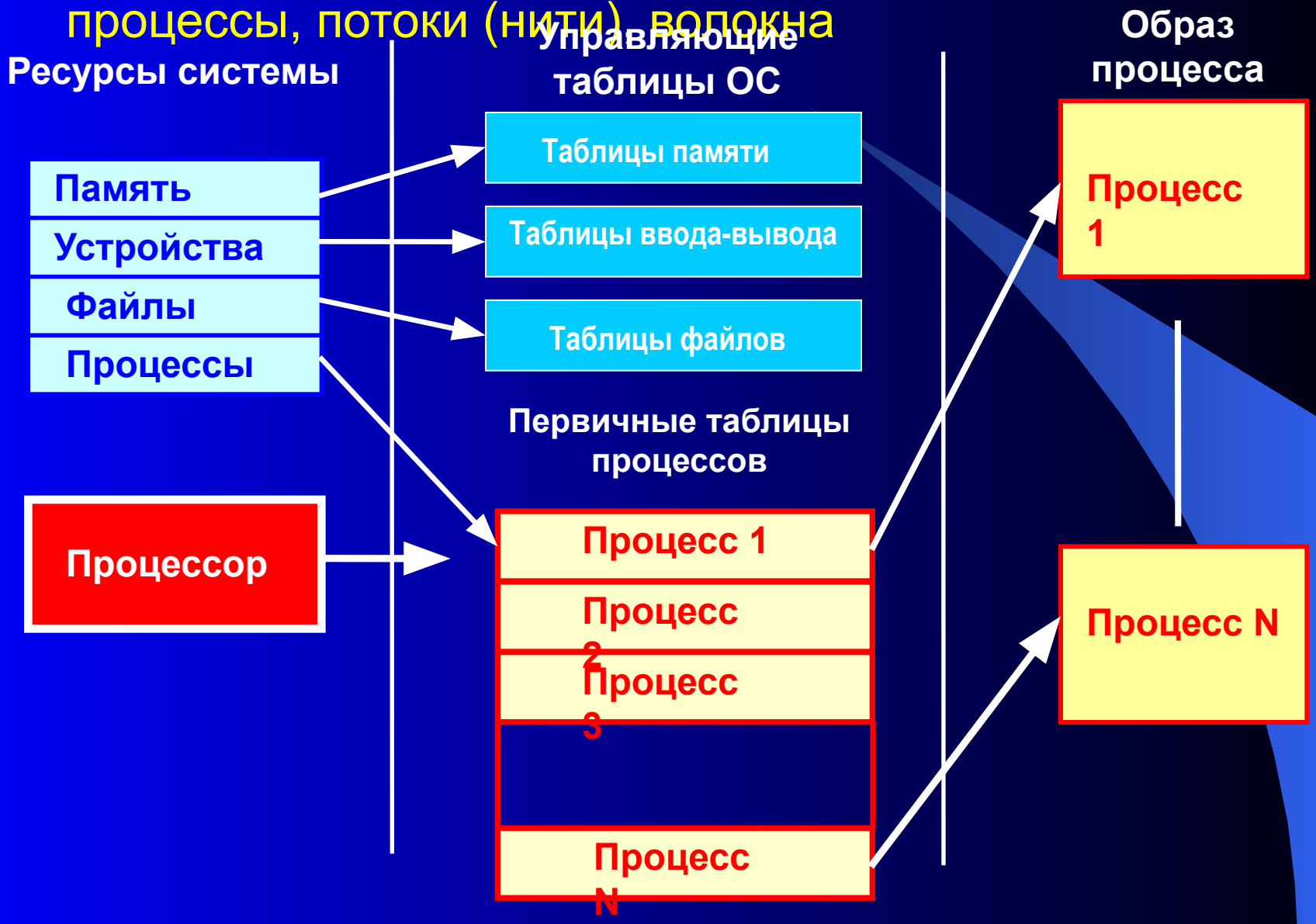
2.4. Создание процессов и потоков. Модели процессов и потоков

2.5. Планирование процессов и потоков

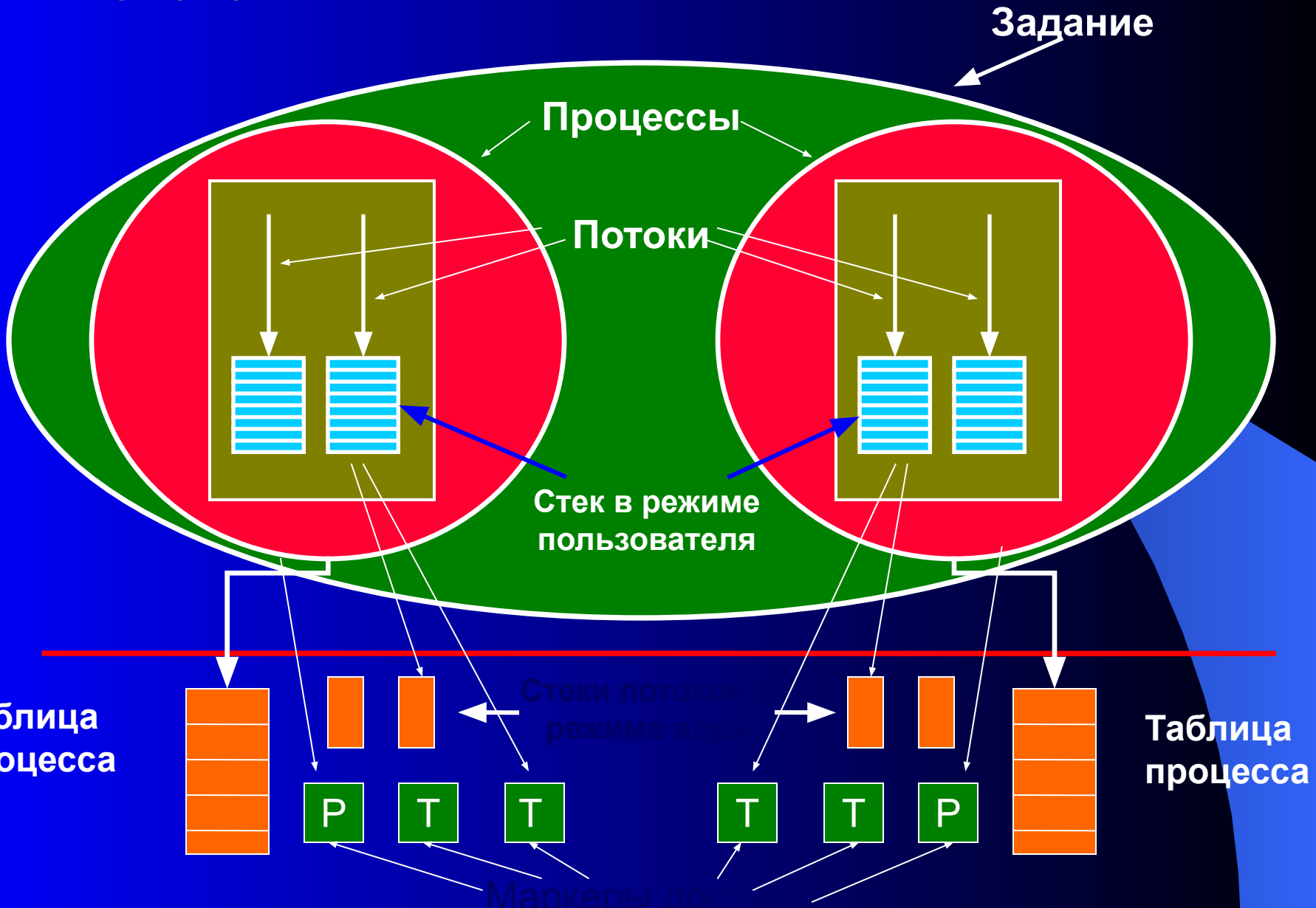
2.6. Взаимодействие и синхронизация процессов и потоков

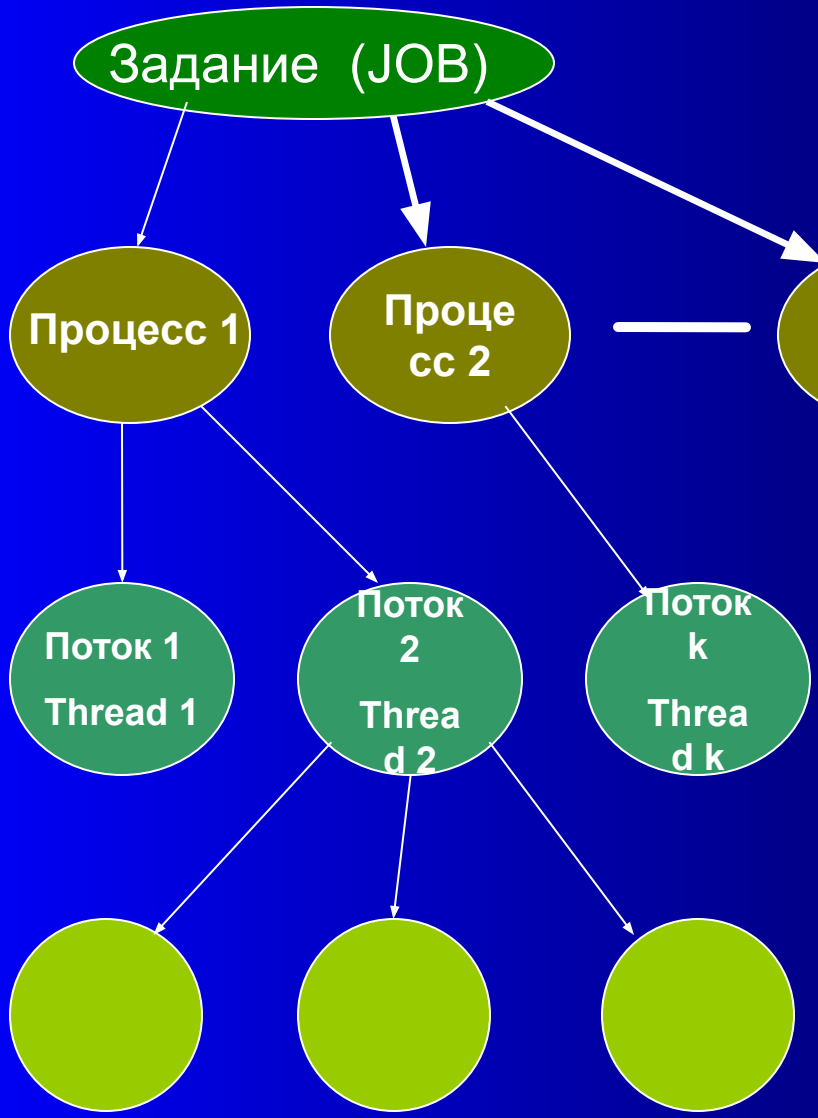
2.7. Аппаратно-программные средства поддержки мультипрограммирования

2.1. Концепция процессов и потоков. Задания, процессы, потоки (нити), волокна



Взаимосвязь между заданиями, процессами и потоками



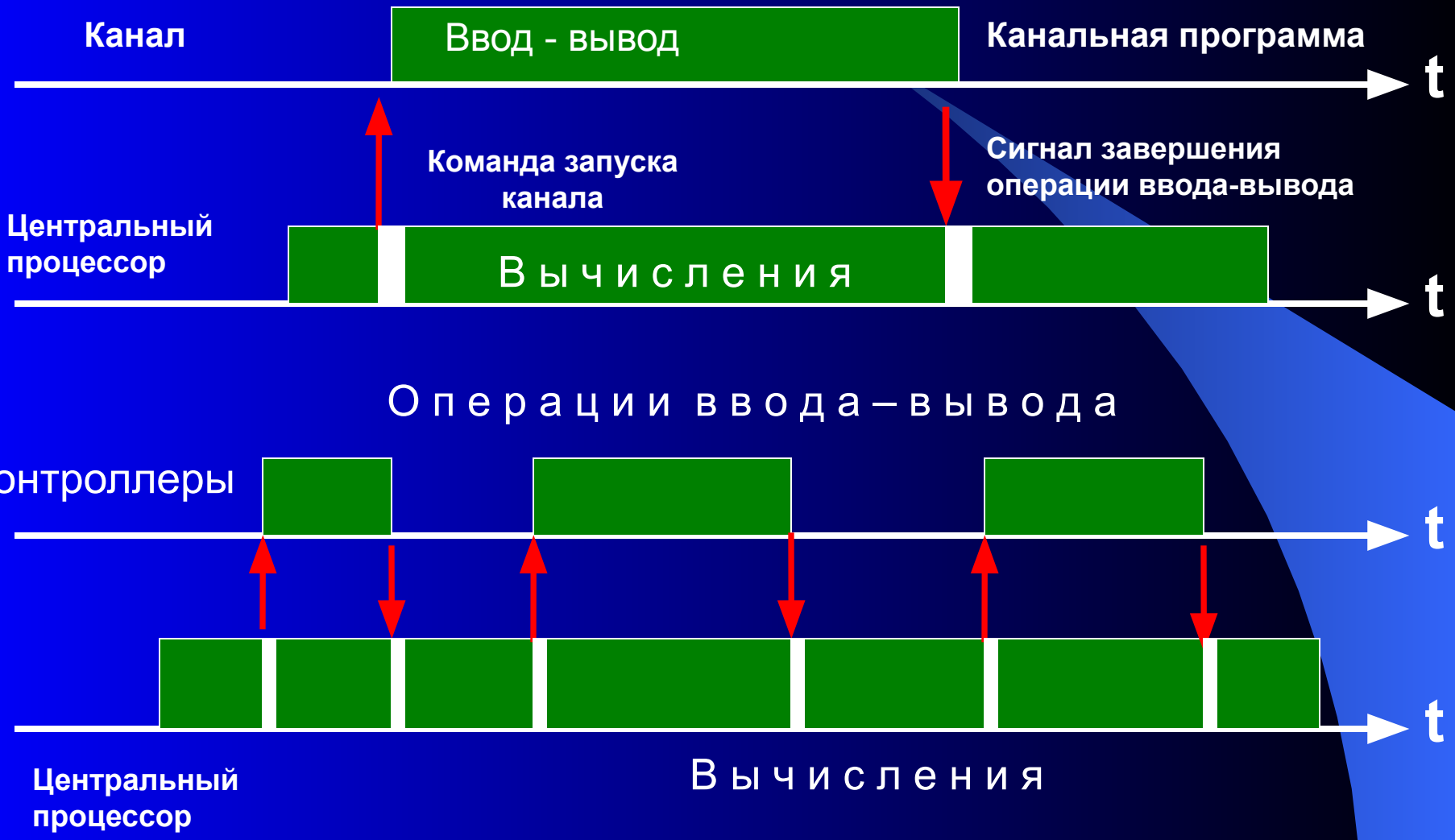


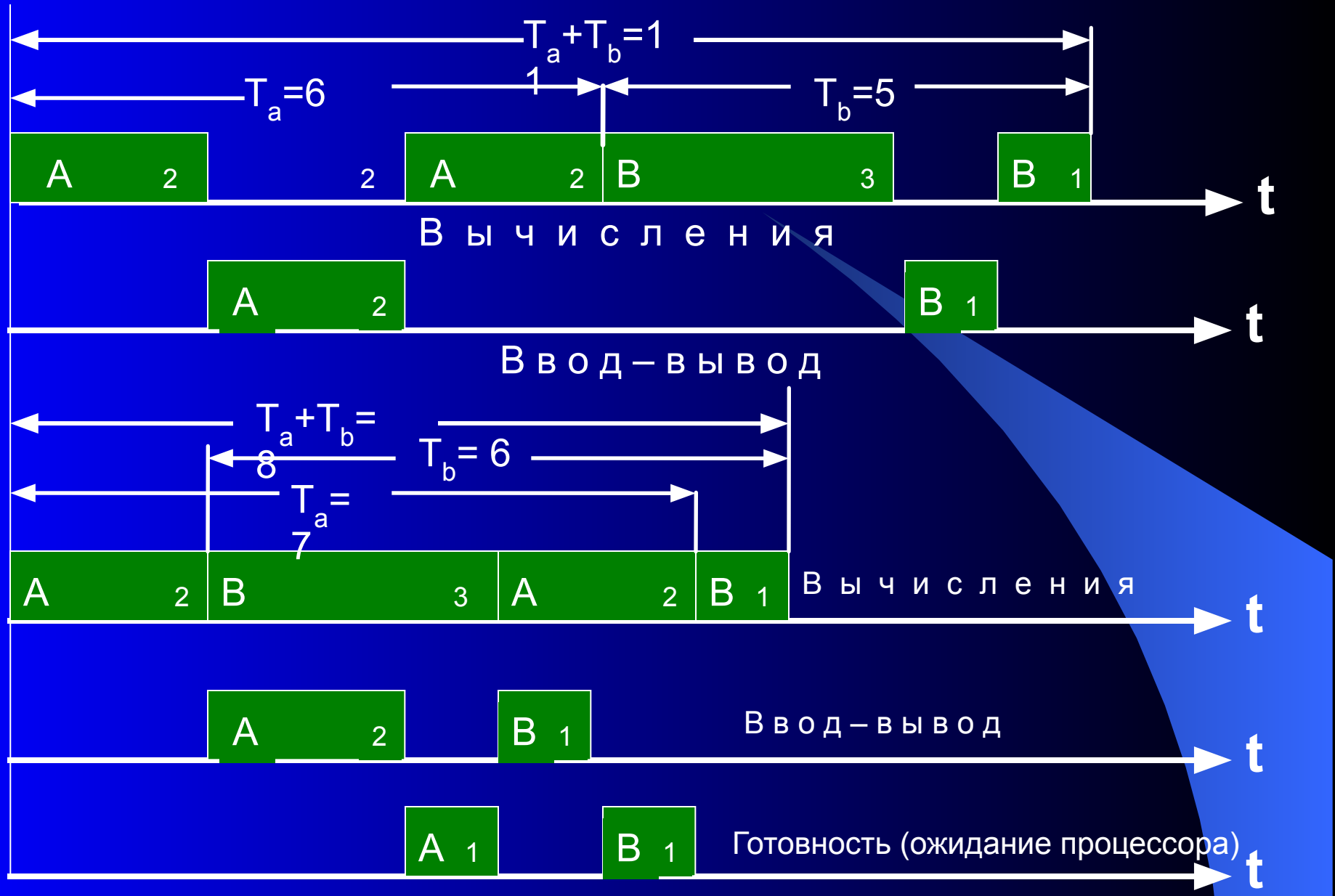
Объекты

Название	Описание
Задание	Набор процессов с общими квотами и лимитами
Процесс	Контейнер для ресурсов и потоков
Поток	Исполнение кода в процессе
Волокно	Облегченный поток, полностью управляемый в пространстве пользователя

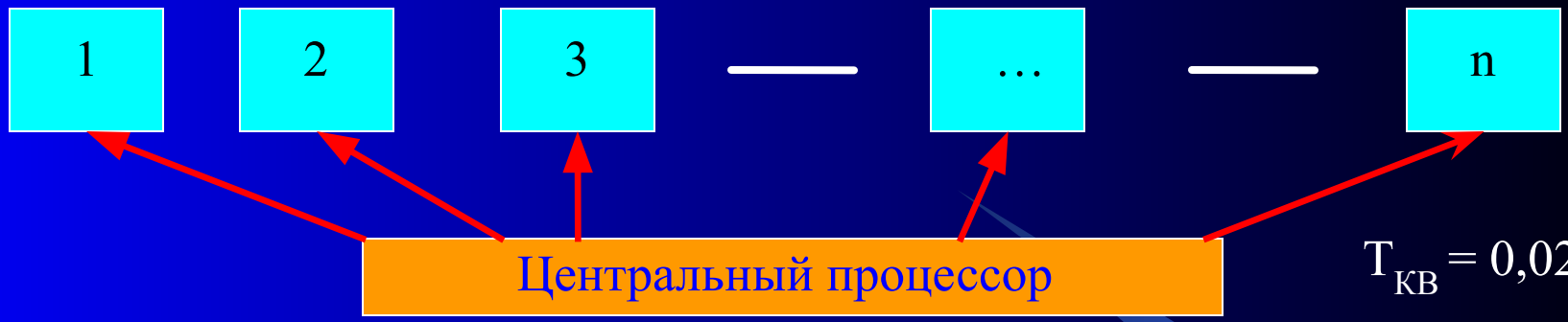
2.2. Мультипрограммирование. Формы многопрограммной работы

2.2.1. Мультипрограммирование в системах пакетной обработки




Ta+Tb=1
Ta=6
1
Tb=5
A 2
2
A 2
B 3
B 1
t
Вычисления
A 2
B 1
t
Ввод–вывод
Ta+Tb=
8
Tb= 6
Ta=
7
A 2
B 3
A 2
B 1
Вычисления
t
A 2
B 1
Ввод–вывод
t
A 1
B 1
Готовность (ожидание процессора)
t

2.2.2. Мультипрограммирование в системах разделения времени



2.2.3. Мультипрограммирование в системах реального времени

1. Управление техническими объектами, технологическими процессами, системами обслуживания и т. п.
2. Фиксированный набор заранее разработанных задач.
3. Жесткие ограничения на время обслуживания.
4. Режим типа запрос – ответ.

5. 2.2.4. Мультипроцессорная обработка

1. Операционные системы : Windows NT/2000/2003, Sun Solaris 2/x, Santa Cruz Operations Open Server 3.x, OS/2 и др.
1. Симметричная архитектура и асимметричная архитектура.

2.3. Управление процессами и потоками

2.3.1. Основные функции управления процессами и потоками

Создание процессов и потоков.

Обеспечение процессов и потоков необходимыми ресурсами.

Изоляция процессов.

Планирование выполнения процессов и потоков.

Диспетчеризация потоков.

Синхронизация процессов и потоков.

Завершение и уничтожение процессов и потоков.

События, приводящие к созданию процессов:

Инициализация (загрузка) ОС.

Запрос процесса на создание дочернего процесса.

Запрос пользователя на создание процесса (например, при входе в систему в интерактивном режиме).

Инициирование пакетного задания.

Создание операционной системой процесса какой-либо службы.

2.3.2. Роль процессов, потоков и волокон в мультипрограммировании

1. Отдельный процесс не может быть выполнен быстрее, чем в однопрограммном режиме.
2. Сложно создать программу, реализующую параллелизм в рамках одного процесса.
3. Стандартные средства современных ОС не позволяют создать для одного приложения несколько процессов для параллельных работ.
4. Многопоточная обработка позволяет распараллелить вычисления в рамках одного процесса.
5. Многопоточная (multithreading) обработка эффективна в многопроцессорных вычислительных системах.
6. Использование аппарата волокон (Windows 2000) повышает эффективность мультипрограммирования за счет сокращения переключения процессов, но увеличивает трудоемкость разработки приложений.

2.4. Создание процессов и потоков. Модели процессов и потоков

Примеры описателей процесса:

блок управления задачей (TCB – Task Control Block) в OS/360;

управляющий блок процесса (PCB – Process Control Block) в OS/2;

дескриптор процесса в UNIX;

объект-процесс (object-process) в Windows NT/2000/2003.

2.4.1.Процессы и их модели

Образ процесса: программа, данные, стек и атрибуты процесса

Информация	**Описание**
Данные пользователя	Изменяемая часть пользовательского адресного пространства (данные программы, пользовательский стек и модифицируемый код)
Пользовательская программа	Программа, которую нужно выполнить
Системный стек	Один или несколько системных стеков для хранения параметров и адресов вызова процедур и системных служб
Управляющий блок процесса	Данные, необходимые ОС для управления процессом: 1) дескриптор процесса, 2) контекст процесса

Дескриптор процесса содержит:

1. Информацию по идентификации процесса (идентификатор процесса, идентификатор пользователя, идентификаторы родительского и дочерних процессов).
2. Информацию по состоянию процесса
3. Информацию, используемую для управления процессом

Информация по состоянию и управлению процессом

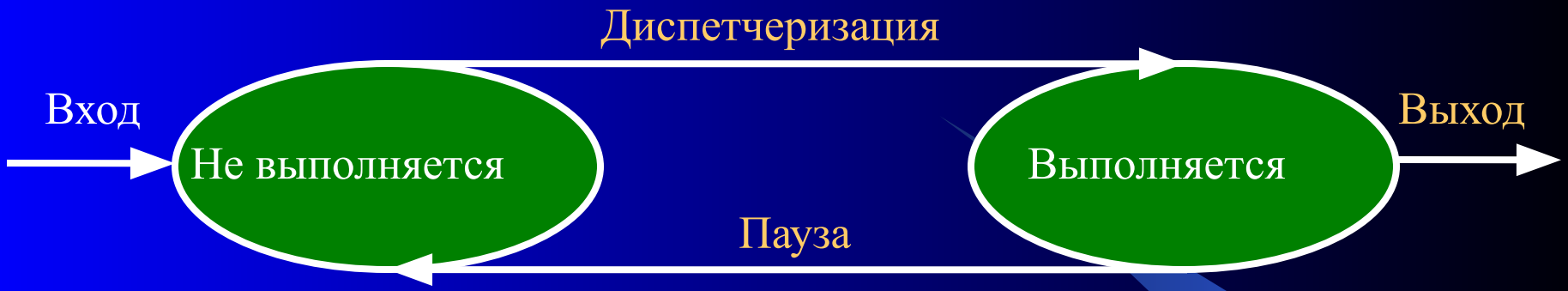
- **Состояние процесса, определяющее его готовность к выполнению (выполняющийся, готовый к выполнению, ожидающий события, приостановленный);**
- **Данные о приоритете (текущий, по умолчанию, максимально возможный);**
- **Информация о событиях – идентификация события, наступление которого позволит продолжить выполнение процесса;**
- **Указатели, позволяющие определить расположение образа процесса в оперативной памяти и на диске;**
- **Указатели на другие процессы (находящиеся в очереди на выполнение);**
- **Флаги, сигналы и сообщения, имеющие отношение к обмену информацией между двумя независимыми процессами;**
- **Данные о привилегиях, определяющие прав доступа к определенной области памяти или возможности выполнять определенные виды команд, использовать системные утилиты и службы;**
- **Указатели на ресурсы, которыми управляет процесс;**
- **Сведения по использованию ресурсов и процессора;**
- **Информация, связанная с планированием.**

КОНТЕКСТ ПРОЦЕССА

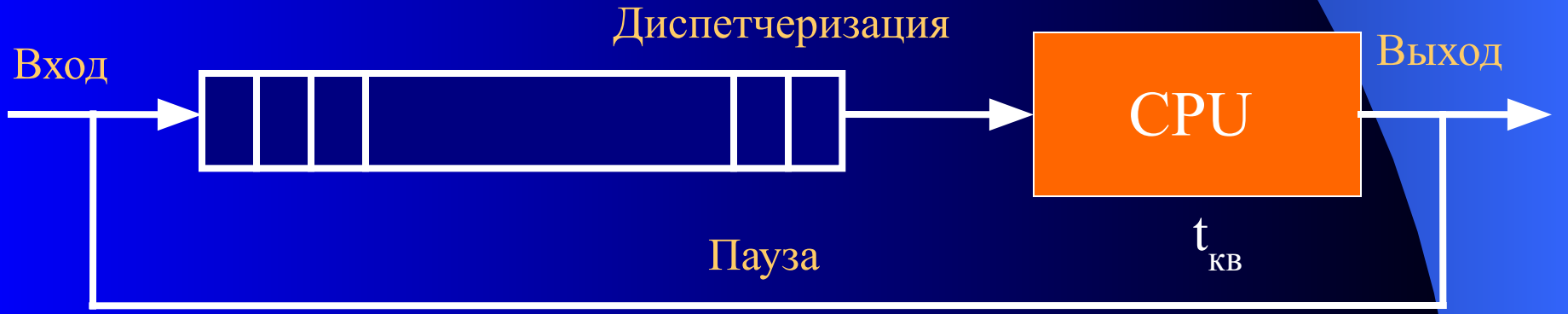
- Содержимое регистров процессора, доступных пользователю (обычно 8 – 32 регистра и до 100 регистров в RISC – процессорах);
- Содержимое счетчика команд;
- Состояние управляющих регистров и регистров состояния;
- Коды условия, отражающие результат выполнения последней арифметической или логической операции (например, равенство нулю, переполнение);
- Указатели вершин стеков, хранящие параметры и адреса вызова процедур и системных служб.

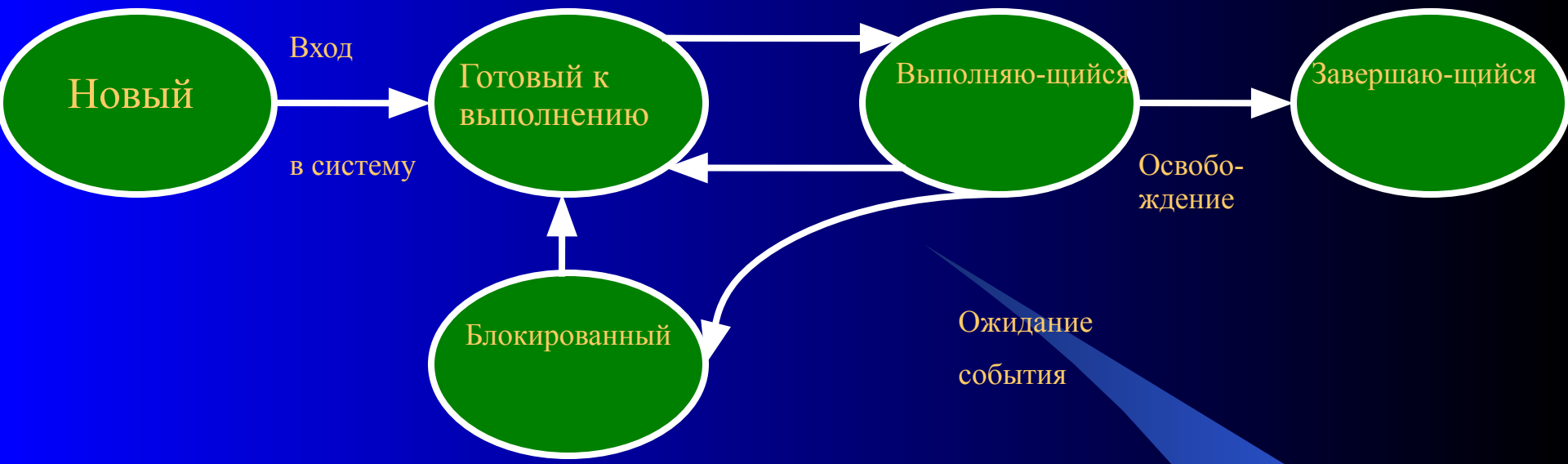
Значительная часть этой информации фиксируется в виде слова состояния программы PSW (program status word – EFLAGS в процессоре Pentium).

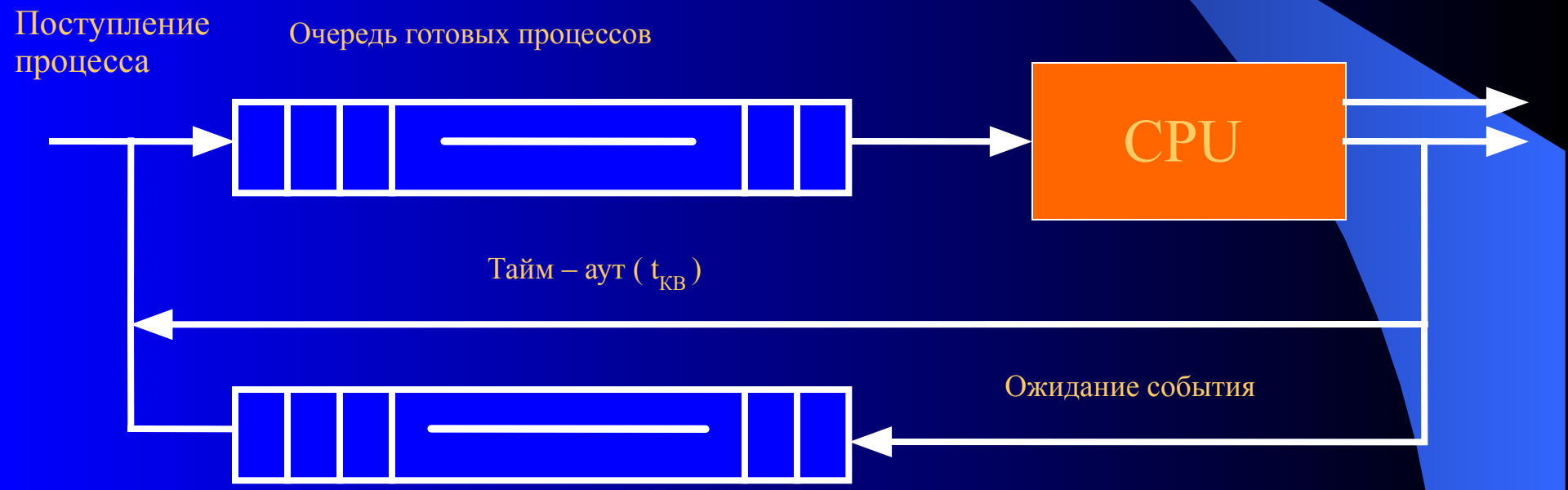
Простейшая модель процесса



Граф состояний и переходов




Новый
Вход
в систему
Готовый к выполнению
Выполняю-щийся
Завершаю-щийся
Освобо-ждение
Блокированный
Ожидание
события


Поступление процесса
Очередь готовых процессов
CPU
Тайм – аут ($t_{КВ}$)
Ожидание события

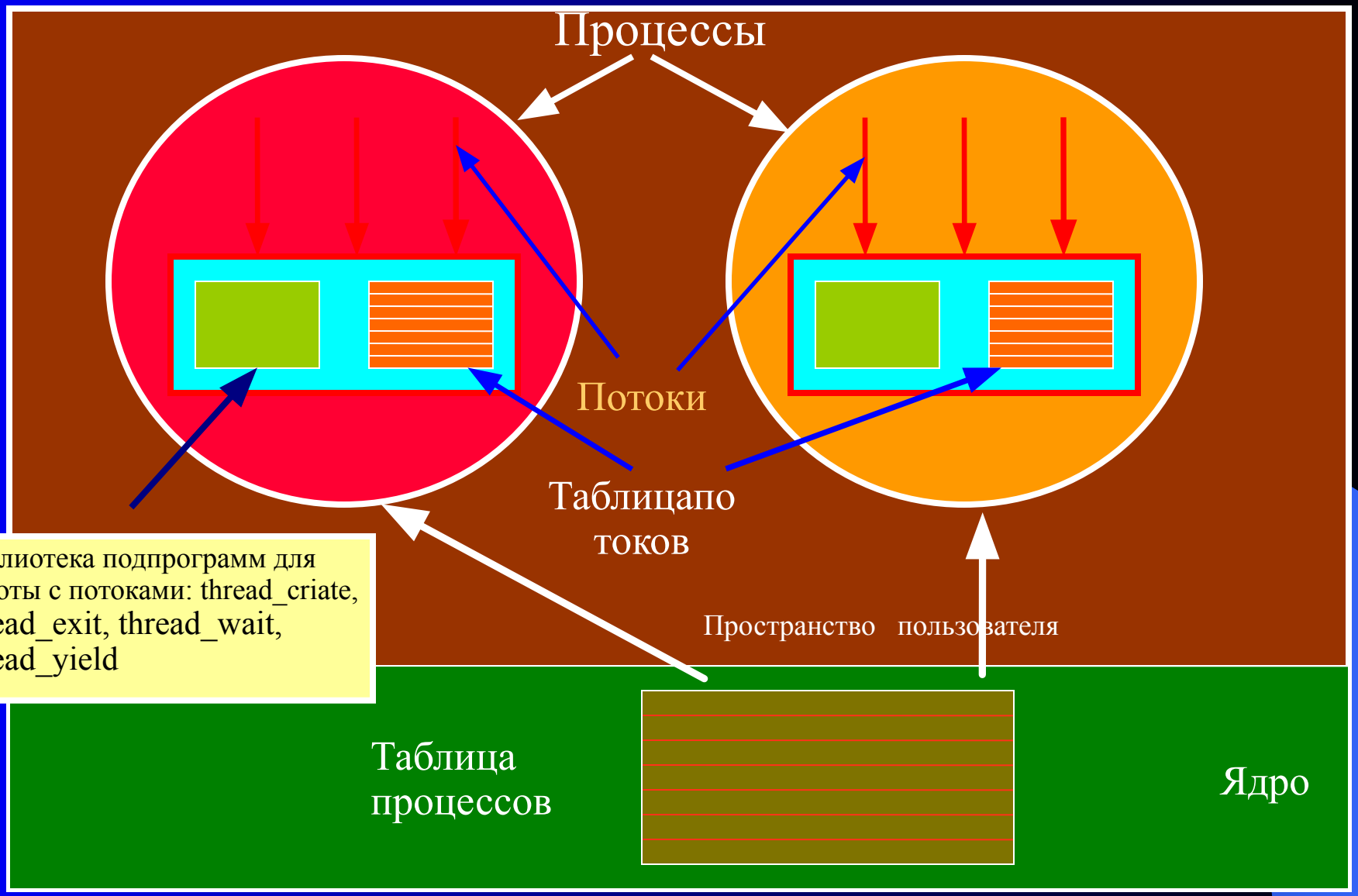
2.4.2. Потоки и их модели

Описатель потока: блок управления потоком и контекст потока (в многопоточной системе процессы контекстов не имеют).

Способы реализации пакета потоков:

- в пространстве пользователя (user – level threads – ULT);
- в ядре (kernel – level threads – KLT).

Поток на уровне пользователя (в пользовательском пространстве)



Поток на уровне пользователя

ДОСТОИНСТВА:

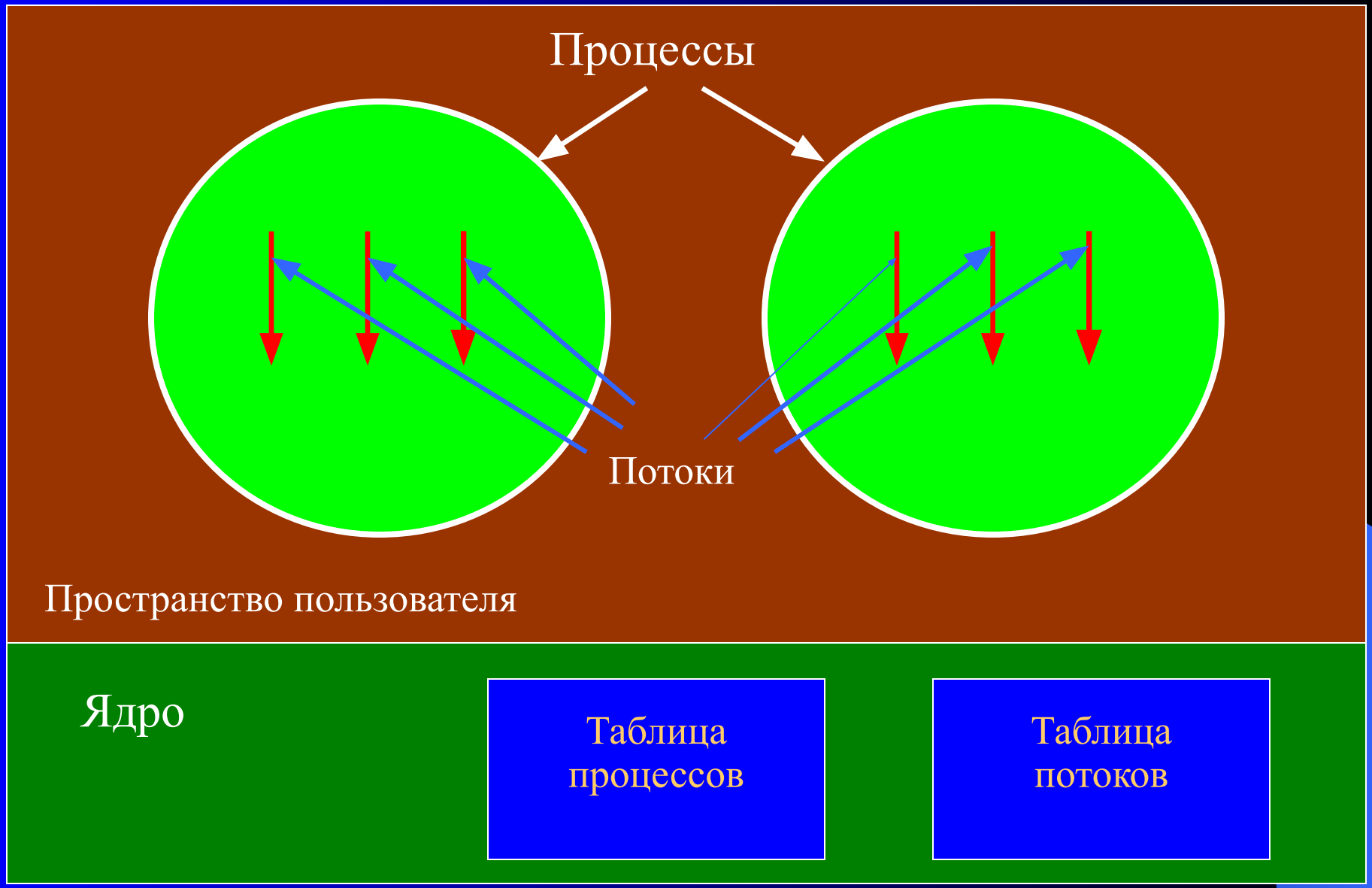
- можно реализовать в ОС, не поддерживающей потоки без каких-либо изменений в ОС;
- высокая производительность, поскольку процессу не нужно переключаться в режим ядра и обратно;
- ядро о потоках ничего не знает и управляет однопоточными процессами;
- имеется возможность использования любых алгоритмов планирования потоков с учетом их специфики;
- управление потоками возлагается на программу пользователя.

Поток на уровне пользователя

НЕДОСТАТКИ:

- системный вызов блокирует не только работающий поток, но и все потоки того процесса, к которому он относится;
- приложение не может работать в многопроцессорном режиме, так как ядро закрепляет за каждым процессом только один процессор;
- при запуске одного потока ни один другой поток а рамках одного процесса не будет запущен пока первый добровольно не отдаст процессор;
- внутри одного потока нет прерываний по таймеру, в результате чего невозможно создать планировщик по таймеру для поочередного выполнения потоков.

Поток на уровне ядра



Поток на уровне ядра

ДОСТОИНСТВА:

- возможно планирование работы нескольких потоков одного и того же процесса на нескольких процессорах;
- реализуется мультипрограммирование в рамках всех процессов (в том числе одного);
- при блокировании одного из потоков процесса ядро может выбрать другой поток этого же (или другого процесса);
- процедуры ядра могут быть многопоточными.

НЕДОСТАТКИ:

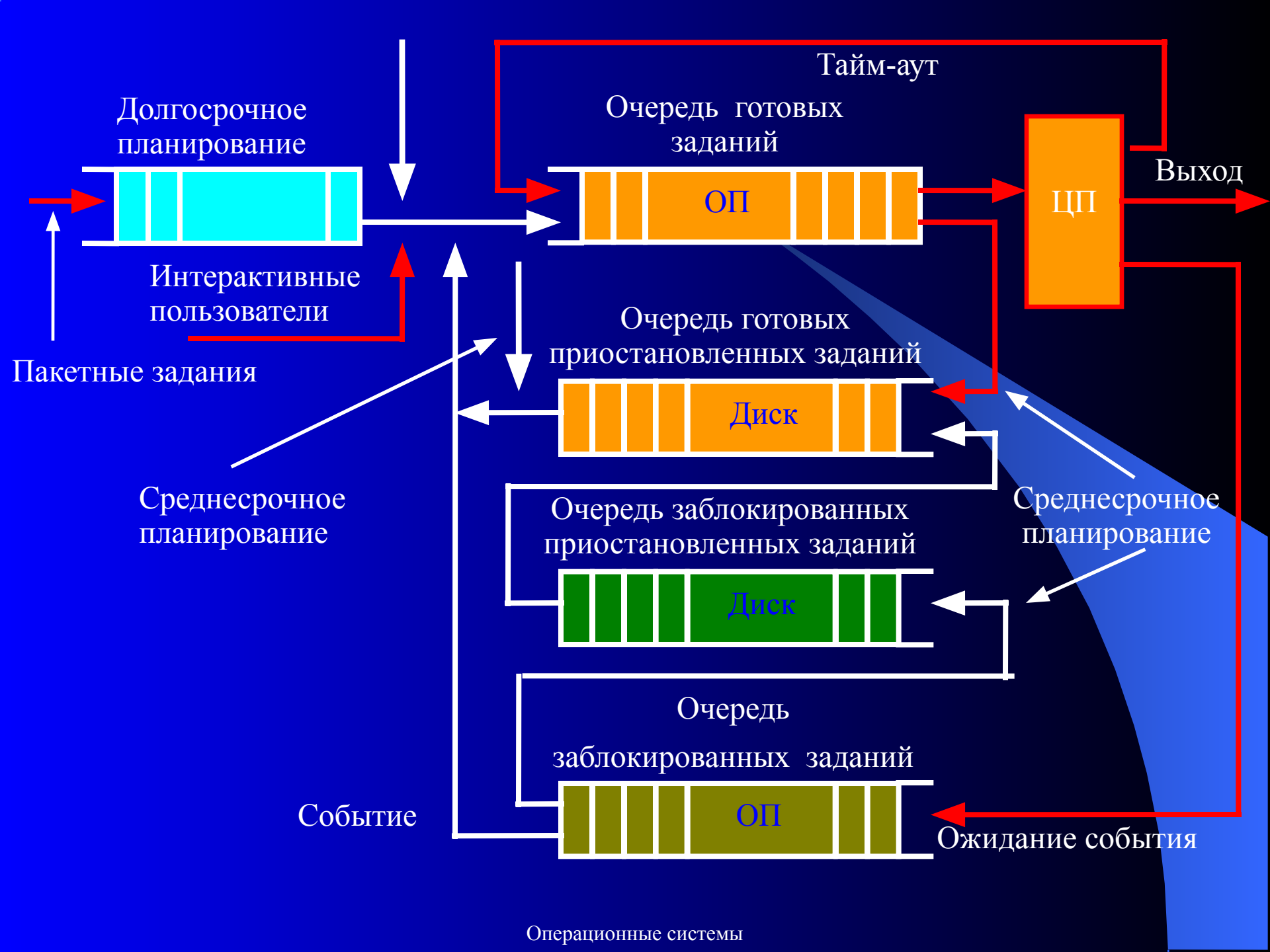
Необходимость двукратного переключения режима пользователь – ядро, ядро – пользователь для передачи управления от одного потока к другому в рамках одного и того же процесса.

2.5. Планирование заданий, процессов и потоков

1. Виды планирования

Вид планирования	Выполняемые функции
Долгосрочное	Решение о добавлении задания (процесса) в пул выполняемых в системе
Среднесрочное	Решение о добавлении процесса к числу процессов полностью или частично размещенных в основной памяти
Краткосрочное	Решение о том, какой из доступных процессов (потоков) будет выполняться процессором
Планирование ввода-вывода	Решение о том, какой из запросов процессов (потоков) на операцию ввода-вывода будет выполняться свободным устройством ввода-вывода

Схема планирования с учетом очередей заданий (процессов)


Тайм-аут
Долгосрочное планирование
Очередь готовых заданий
ОП
ЦП
Выход
Интерактивные пользователи
Пакетные задания
Очередь готовых приостановленных заданий
Диск
Среднесрочное планирование
Очередь заблокированных приостановленных заданий
Диск
Среднесрочное планирование
Очередь заблокированных заданий
ОП
Событие
Ожидание события

Типичный граф состояния потока

Алгоритмы планирования потоков

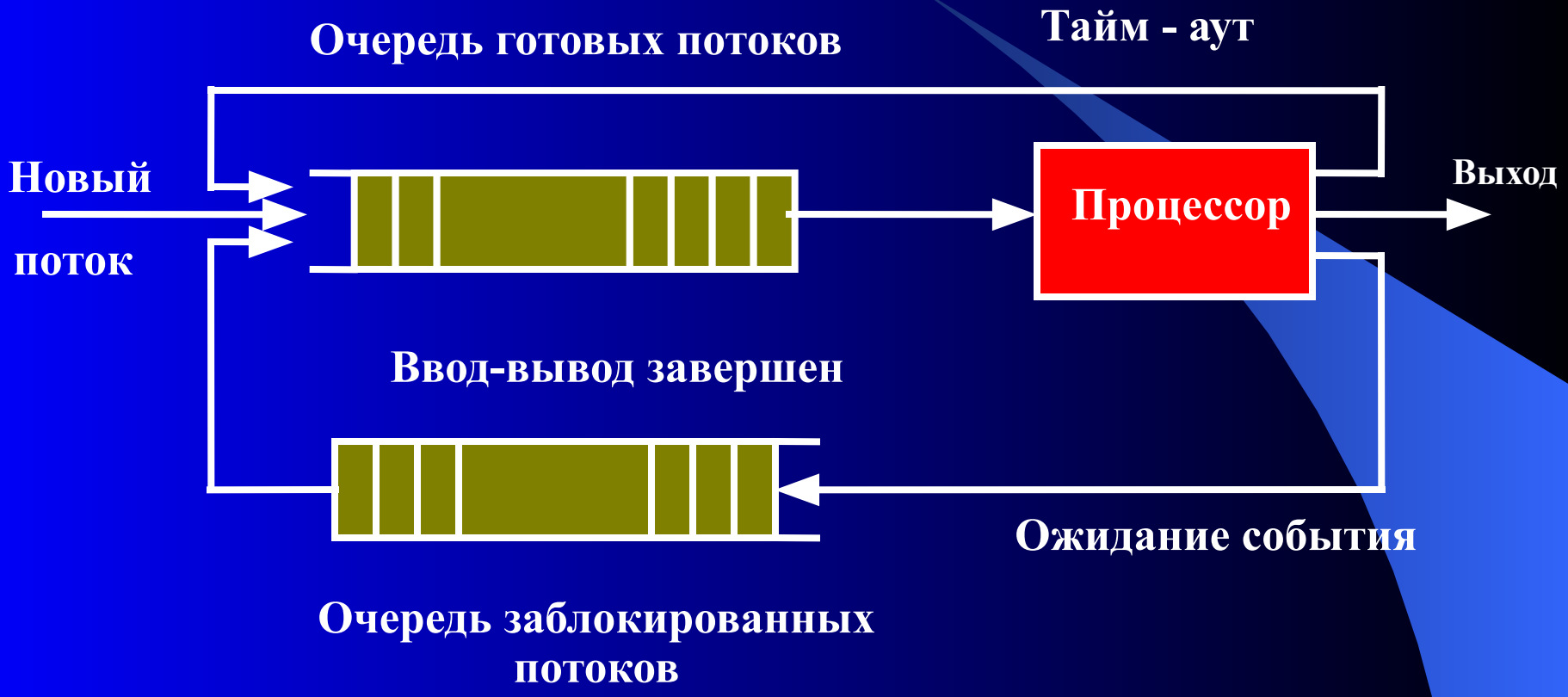
1. Невытесняющие (non-preemptive)

- планирование распределяется между ОС и прикладными программами;
- необходимость частых передач управлений ОС, в противном случае возможна монополизация процессора приложением;
- зависания приложений могут привести к краху системы

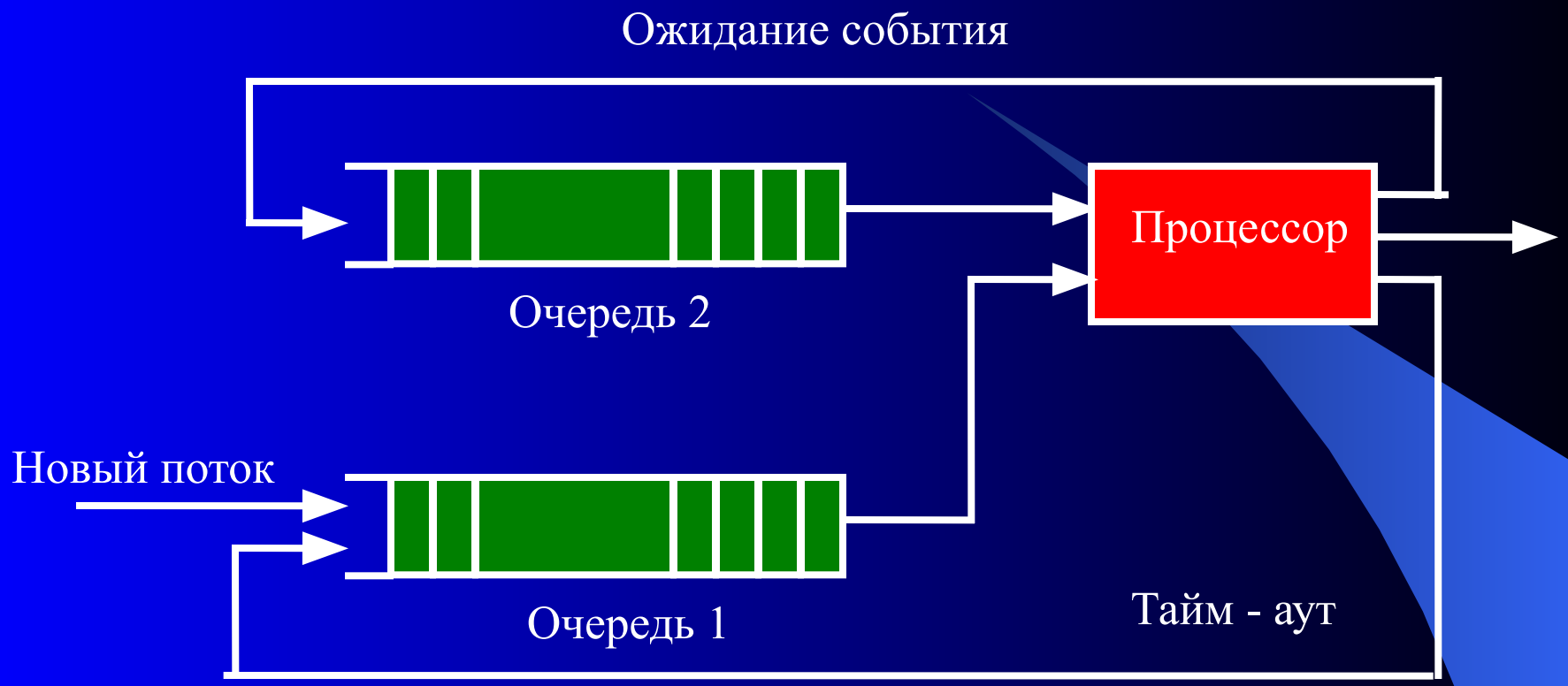
2. Вытесняющие (preemptive)

- функции планирования сосредоточены в ОС;
- планирование на основе квантования процессорного времени;
- планирование на основе приоритетов потоков: статических, динамических, абсолютных, относительных, смешанных;

Простейший алгоритм планирования

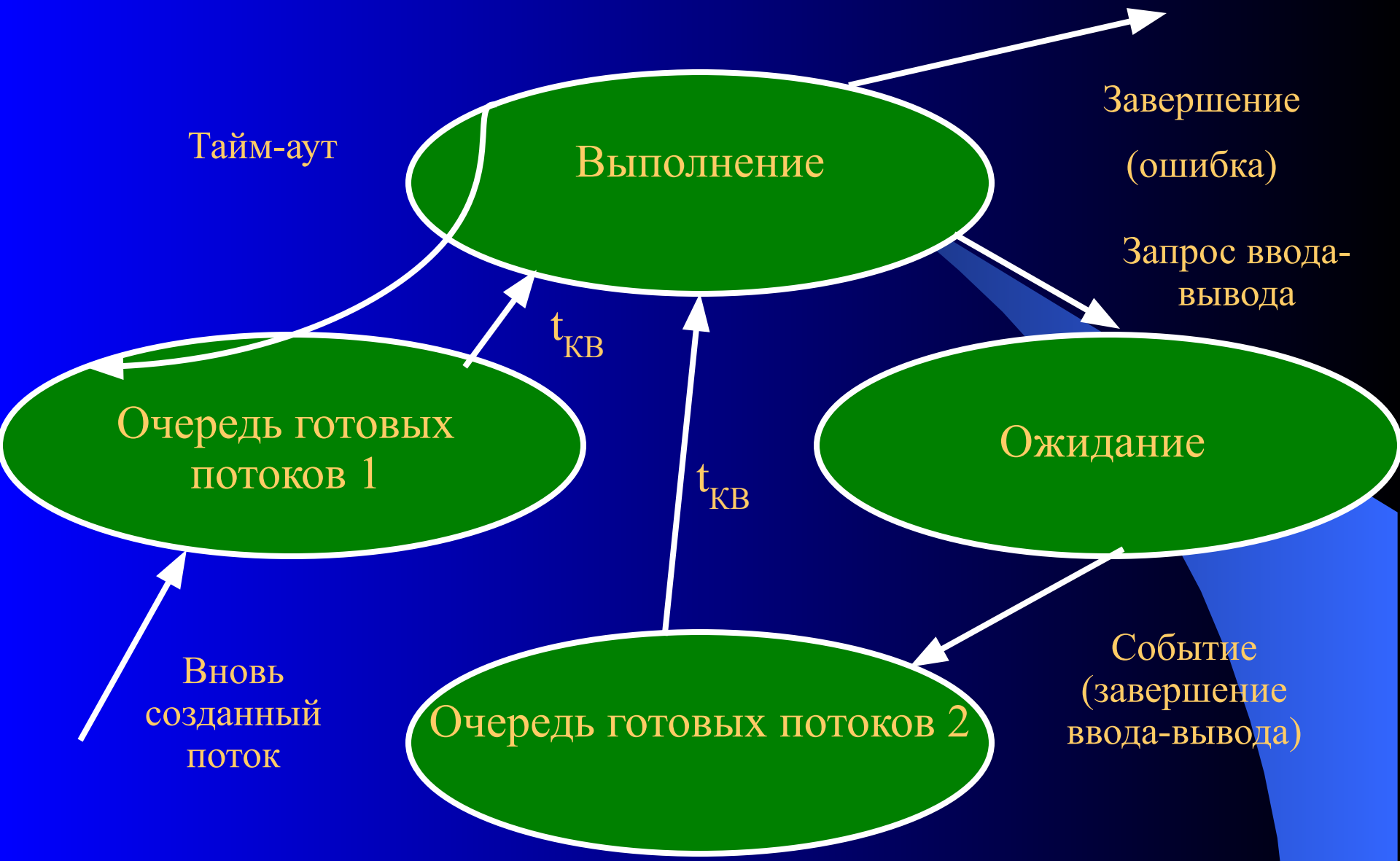


Алгоритм планирования, реализующий предпочтения потокам с интенсивным вводом-выводом

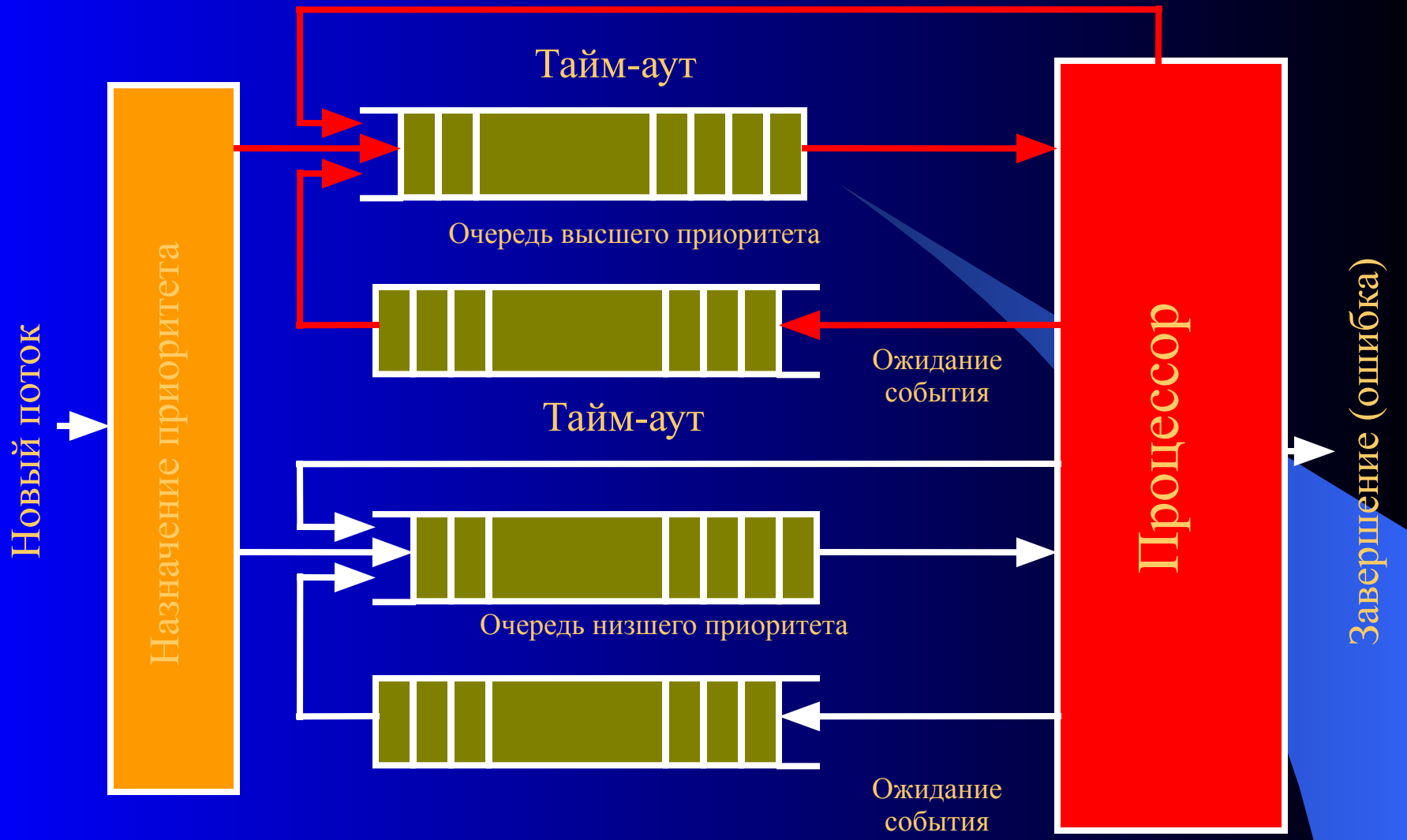


1. Переключение контекстов потоков связано с потерями процессорного времени.
2. С увеличением времени кванта ухудшается обслуживание пользователей.
3. В алгоритмах, основанных на квантовании, ОС не имеет никаких сведений о характеристиках решаемых задач.

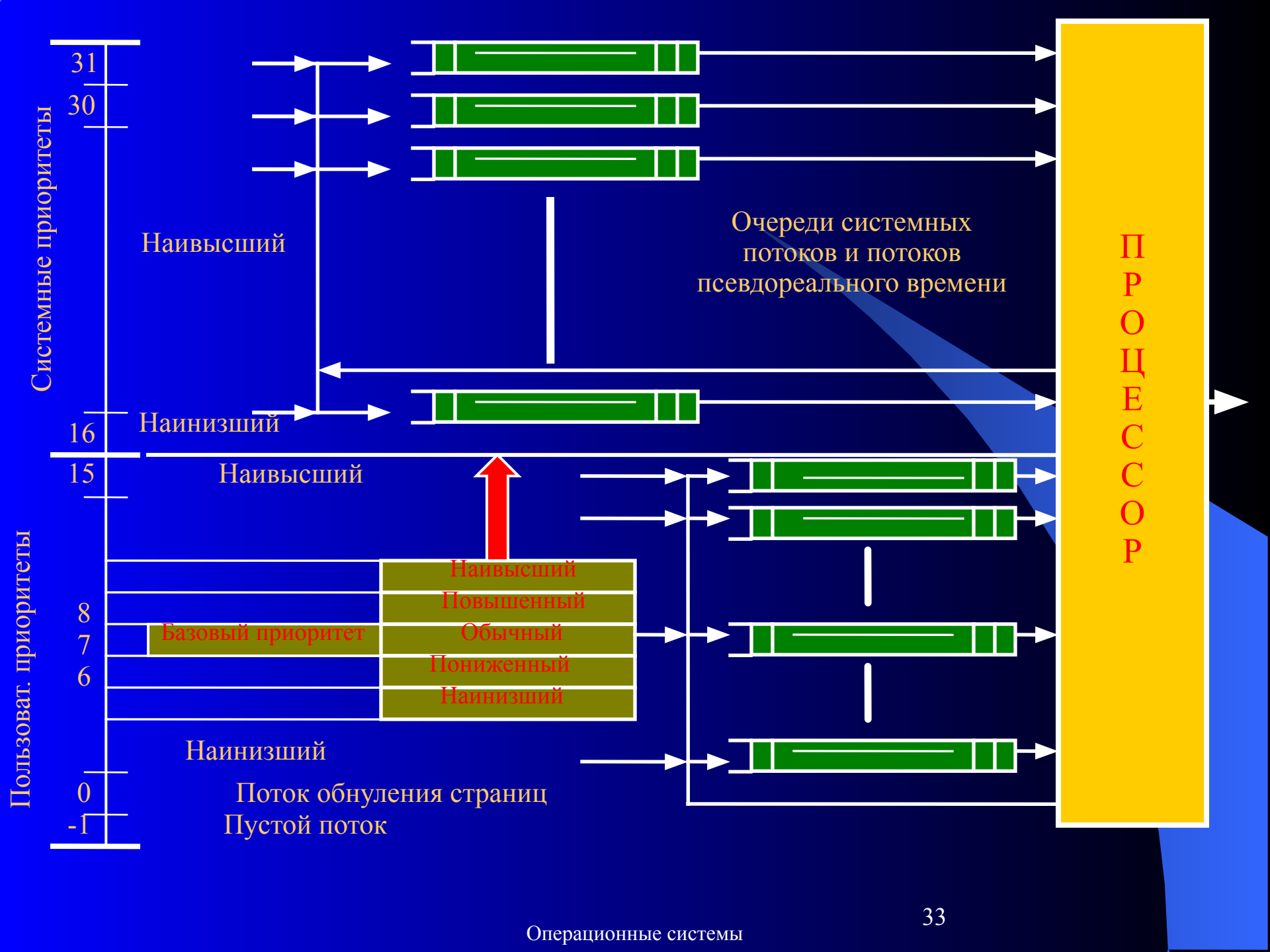
Граф состояния потока



Алгоритмы приоритетного планирования



Приоритетное переключение с квантованием


Системные приоритеты
31
30
Наивысший
Очереди системных потоков и потоков псевдореального времени
ПРОЦЕССОР
Наинизший
16
15
Наивысший
Пользоват. приоритеты
Наивысший
Повышенный
8
7
Базовый приоритет
Обычный
Пониженный
6
Наинизший
Наинизший
0
Поток обнуления страниц
-1
Пустой поток

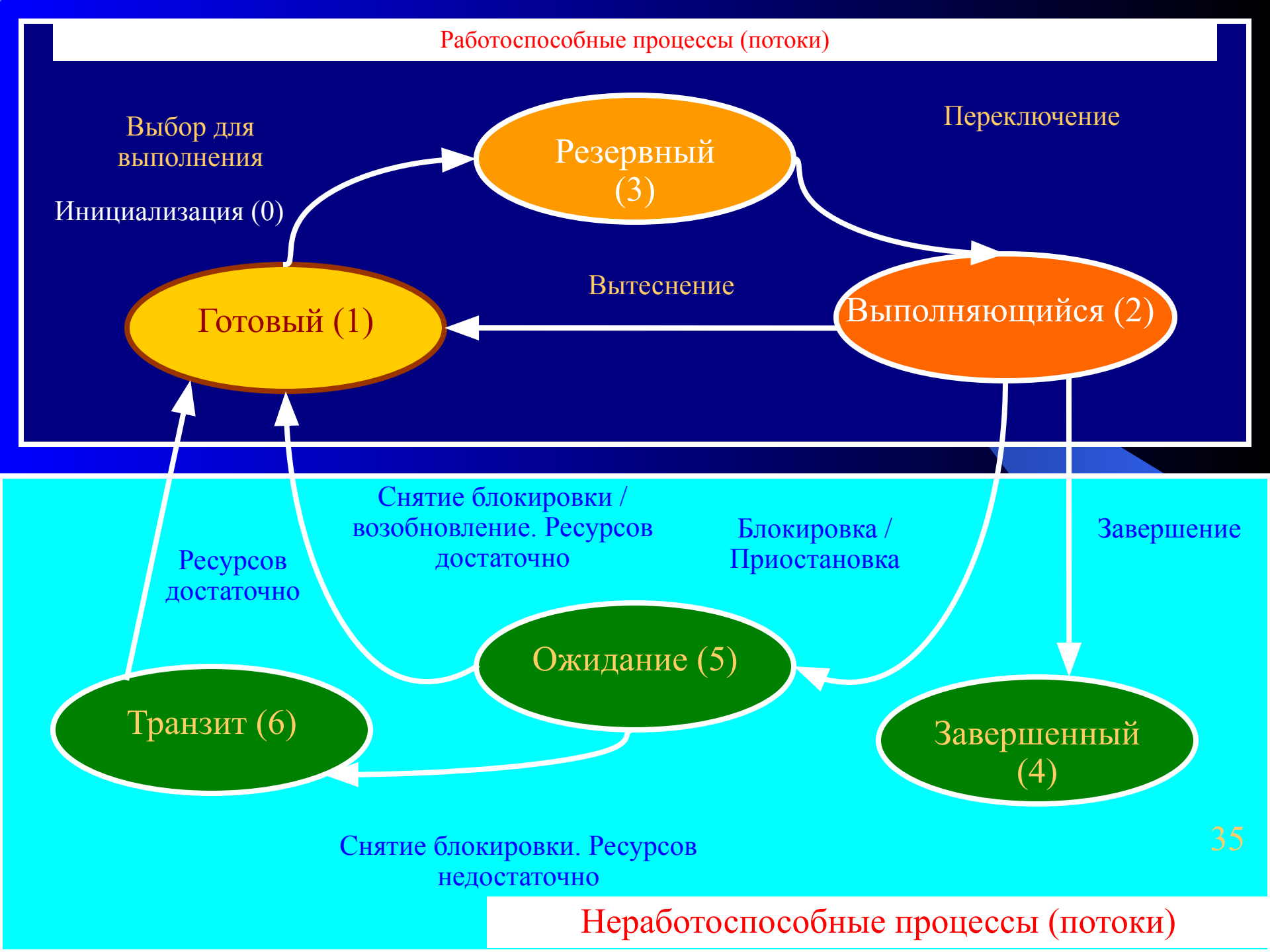
Изменение базового приоритета потока

Увеличение приоритета

+ 1 – завершение ввода-вывода по диску;
+ 2 – для последовательной линии;
+ 6 – клавиатура;
+ 8 – звуковая карта;
+ 2 – снимается блокировка по семафору (для потока переднего плана);
+ 1 - снимается блокировка по семафору (для потока непереднего плана); приоритет 15 на 2 кванта процессора, если готовый к выполнению поток простаивает более некоторого директивного времени.

Уменьшение приоритета

- 1 – если полностью использован квант времени процессора (многократно, вплоть до базового приоритета).


Работоспособные процессы (потоки)
Выбор для выполнения
Инициализация (0)
Резервный (3)
Переключение
Готовый (1)
Вытеснение
Выполняющийся (2)
Снятие блокировки / возобновление. Ресурсов достаточно
Блокировка / Приостановка
Завершение
Ресурсов достаточно
Ожидание (5)
Транзит (6)
Завершенный (4)
Снятие блокировки. Ресурсов недостаточно
Неработоспособные процессы (потоки)

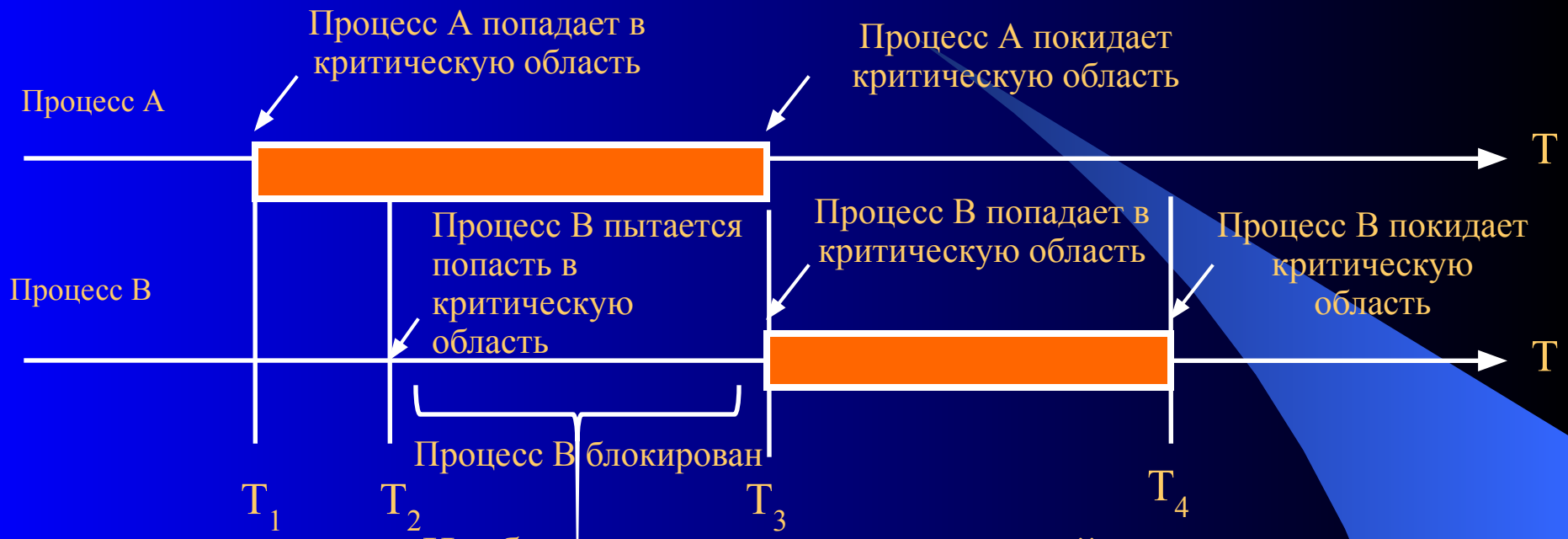
2.6. Взаимодействие и синхронизация процессов и потоков

2.6.1. Проблемы взаимодействия и синхронизации

Степень осведомленности	Взаимосвязь	Влияние одного процесса на другой	Потенциальные проблемы
Процессы не осведомлены друг о друге	Конкуренция	• Результат работы одного процесса не зависит от действий других. • Возможно влияние одного процесса на время работы другого.	• Взаимоисключения • Взаимоблокировки • Голодание
Процессы косвенно осведомлены о наличии друг друга	Сотрудничество с использованием разделения	• Возможно влияние одного процесса на время работы другого. • Результат работы одного процесса может зависеть от информации, полученной от других.	• Взаимоисключения • Взаимоблокировки • Голодание • Синхронизация
Процессы непосредственно осведомлены о наличии друг друга	Сотрудничество с использованием связи	• Результат работы одного процесса зависит от информации, полученной от других процессов. • Возможно влияние одного процесса на время работы другого.	• Взаимоблокировки (расходуемые ресурсы) • Голодание

2.6.2. Конкуренция процессов в борьбе за ресурсы

<u>Конкуренция</u> – ситуация, когда два или более процессов требуют доступ к одному и тому же ресурсу (принтеру, файлу и т.п.), называемому <u>критическим</u>. Часть программы, использующая критический ресурс, называется <u>критической секцией.</u>



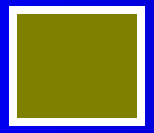
Необходимость взаимоисключений:

1. Процессы не должны одновременно находиться в критических областях.
2. В программе не должно быть предположений о скорости или количестве процессов.
3. Процесс, находящийся вне критической области, не может блокировать другие процессы.
4. Невозможна ситуация, в которой процесс вечно ждет попадания в критическую область.

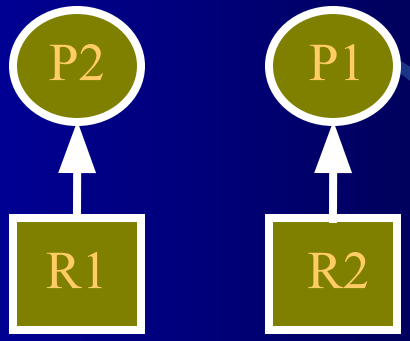
Взаимоблокировки (тупики, deadlock)

Группа процессов находится в **тупиковой ситуации**, если каждый процесс из группы ожидает события, которое может вызвать только другой процесс из этой же группы

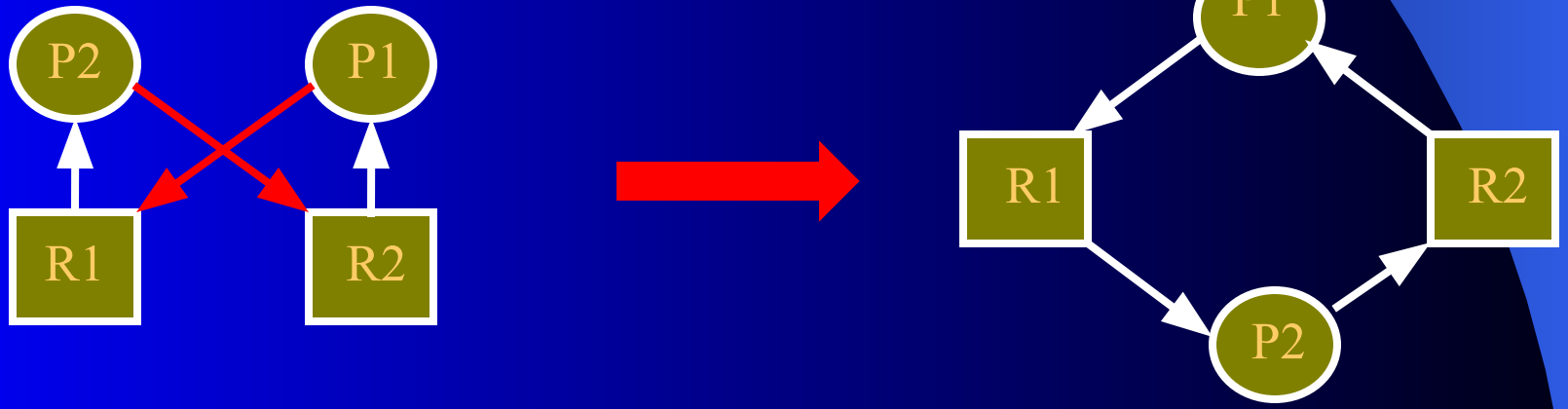
Процесс



Ресурс

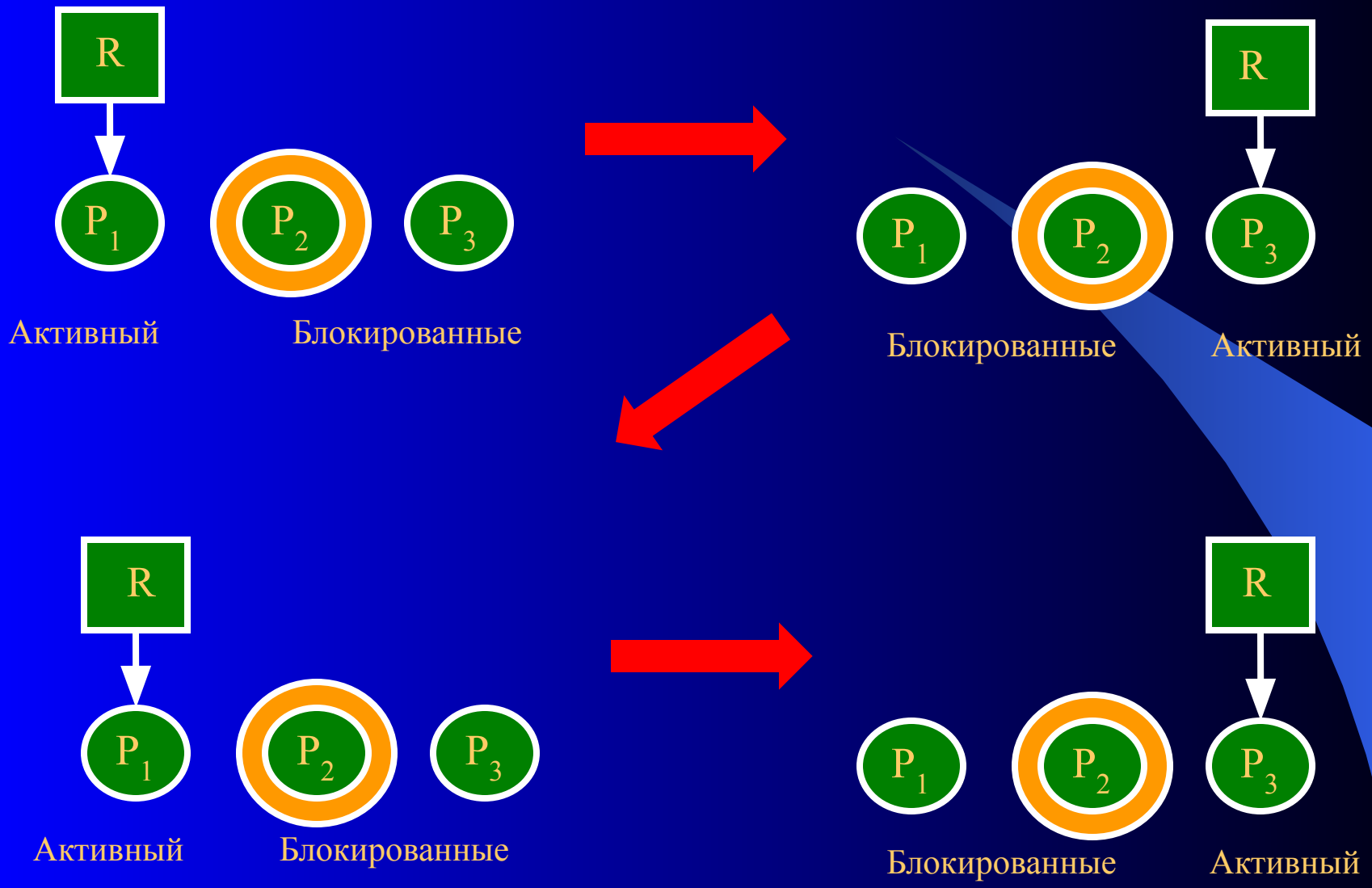


Исходное распределение ресурсов



Тупиковая ситуация

Проблема “голодание”



2.6.3. Сотрудничество с использованием разделения

Процессы, взаимодействующие с другими процессами без наличия явной информации о друг друге, обращаются к разделяемым переменным, к совместно используемым файлам или базам данных.

Проблемы: взаимоисключение, взаимоблокировка, голодание.
Дополнительно: синхронизация процессов для обеспечения согласованности данных

Пример: пусть должно выполняться a = b при начальном значении a = b = 1

1-й вариант: процессы выполняются последовательно

P1: a = a + 1; b = b + 1; P2: b = 2 * b; a = 2 * a;

2-й вариант: процессы прерывают друг друга

P1: a = a + 1; прерывание; P2: b = 2 * b; прерывание;

P1: b = b + 1; прерывание; P2: a = 2 * a;

Согласование нарушено: a = 4, b = 3

Ситуации, в которых два или более процессов обрабатывают разделяемые данные (файлы) и конечный результат зависит от скоростей процессов (потоков), называются гонками.

2.6.4. Методы взаимоисключений

1. Запрещение прерываний при входе в критическую область и разрешение прерываний после выхода из критической области. Достоинства: простота реализации. Недостатки: монополизация процессора, возможный крах ОС при сбое процесса, невозможность использования в многопроцессорных системах.

2. Блокирующие переменные (программный подход)

Недостатки: необходимость постоянного опроса другими потоками, требующими тот же ресурс, блокирующей переменной; дополнительные затраты процессорного времени.