Лекция №11

Управление процессами

(часть 2)

Планирование потоков

Планирование потоков

Планирование – определение момента прерывания текущего активного потока и выбор следующего выполняемого потока из очереди.

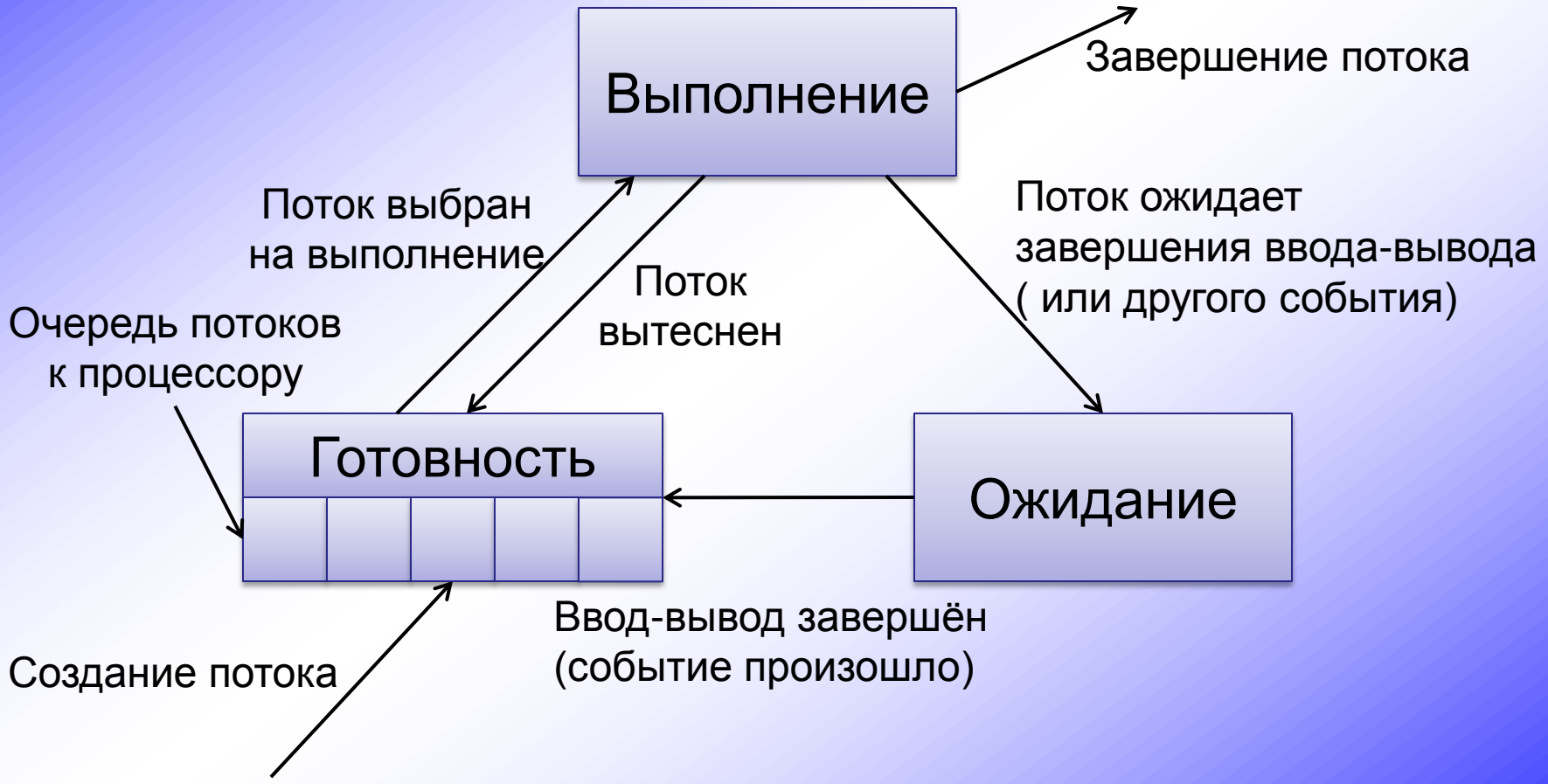
Динамическое планирование – планирование на основе анализа текущей ситуации.

Статическое планирование – планирование на основе расписания.

Состояния потока

- Выполнение – активное состояние потока, во время которого поток обладает всеми необходимыми ресурсами и выполняется процессором.
- Ожидание (блокировка) – пассивное состояние потока, обусловленное ожиданием какого-либо внешнего события.
- Готовность – пассивное состояние потока, обладающего всеми необходимыми ресурсами, во время обработки процессором другого потока.

Граф состояний потока



Моменты перепланировки

Перепланировка – изменение состояния потоков и пересмотр очереди в связи с каким-либо событием, происшедшем в системе.

События, приводящие к перепланировке

- Создание потока (перевод в «готовность»).
- Завершение потока (снятие задачи).
- Изменение приоритета потока.
- Прерывание от таймера об окончании отведённого потоку кванта (перевод из «выполнения» в «готовность»).
- Системный вызов с запросом активной задачи на занятый ресурс (перевод из «выполнения» в «ожидание»).

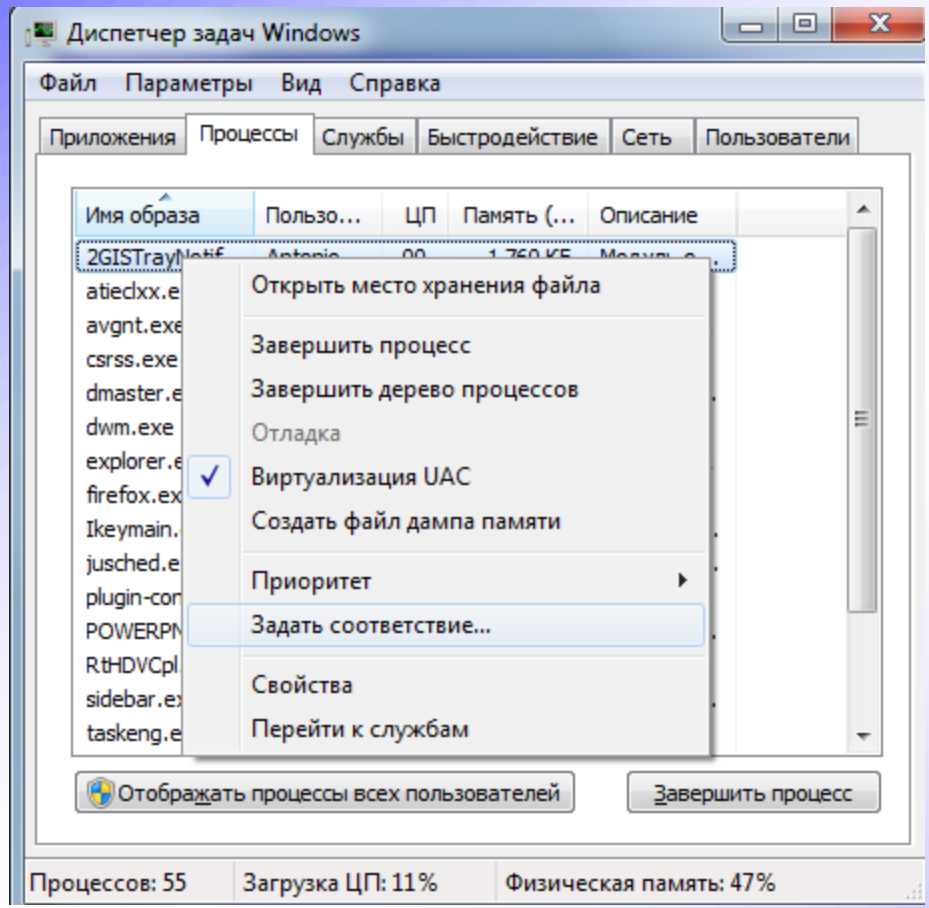
События, приводящие к перепланировке

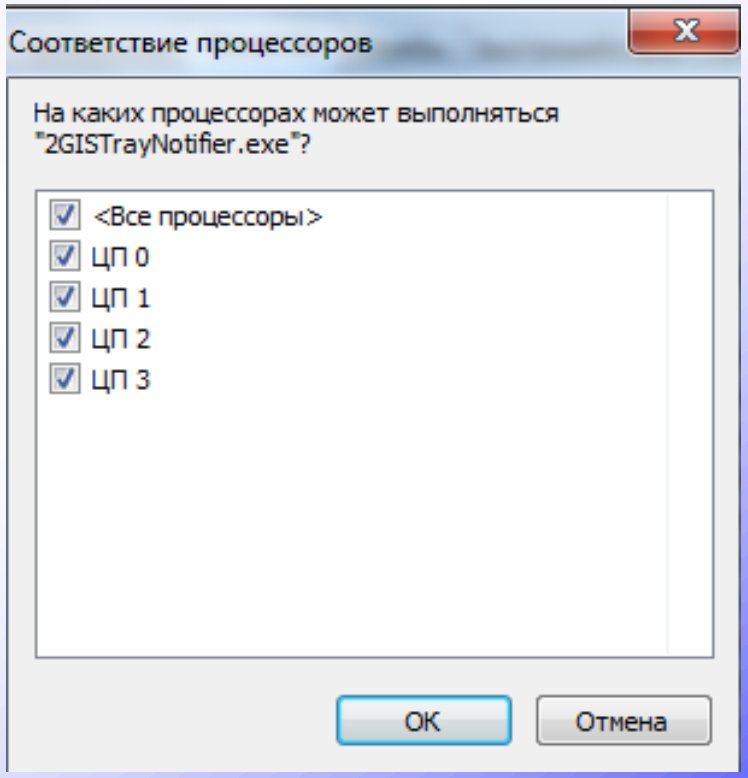
- Системный вызов с освобождением ресурса активной задачей (перевод из «ожидания» в «готовность» ожидающего ресурс потока).
- Внешнее прерывание о завершении устройством операции ввода-вывода (перевод из «ожидания» в «готовность» ожидающего ресурс потока).
- Внутреннее прерывание об ошибке при выполнении активной задачи (снятие задачи).

Привязка потоков к процессору

- По умолчанию в ОС используется нежесткая привязка потоков к процессорам – при прочих равных условиях система пытается выполнять поток на том же процессоре, на котором он работал в последний раз.
- Преимущество – возможность повторного использования данных, сохранившихся в кэше процессора.

Привязка потоков к процессору





Вытесняющие и невытесняющие алгоритмы планирования

- Невытесняющие – активный поток выполняется до тех пор, пока не передаст управление ОС (функции планирования разделены между ОС и приложениями).
- Вытесняющие – переключением процессора с одного потока на другой управляет ОС (планирование осуществляется только ОС).

Особенности невытесняющих алгоритмов планирования

- На время выполнения потока теряется управление системой для пользователя.
- Разработка части функций планировщика ложится на программиста.
- Более простое решение проблем совместного использования ресурсов.
- Более высокая скорость переключения с потока на поток.

Действия при переключении потоков

- Сохранение контекста текущего потока.
- Загрузка контекста выбранного следующим потока.
- Запуск выбранного потока на выполнение.

Факторы, влияющие на планирование

- Затраты времени на переключение процессов.
- Количество процессов, выполняющих вычисления (ограниченных возможностями процессора).
- Количество процессов, ограниченных возможностями устройств ввода-вывода.

Стратегии планирования

Стратегия планирования определяет, какие процессы планируются на выполнение для достижения поставленной цели:

- по возможности заканчивать вычисления (процессы) в порядке их создания;
- отдавать предпочтение более коротким вычислительным задачам;
- предоставлять всем пользователям (их процессам) одинаковые услуги, в том числе и одинаковое время ожидания.

Дисциплины планирования

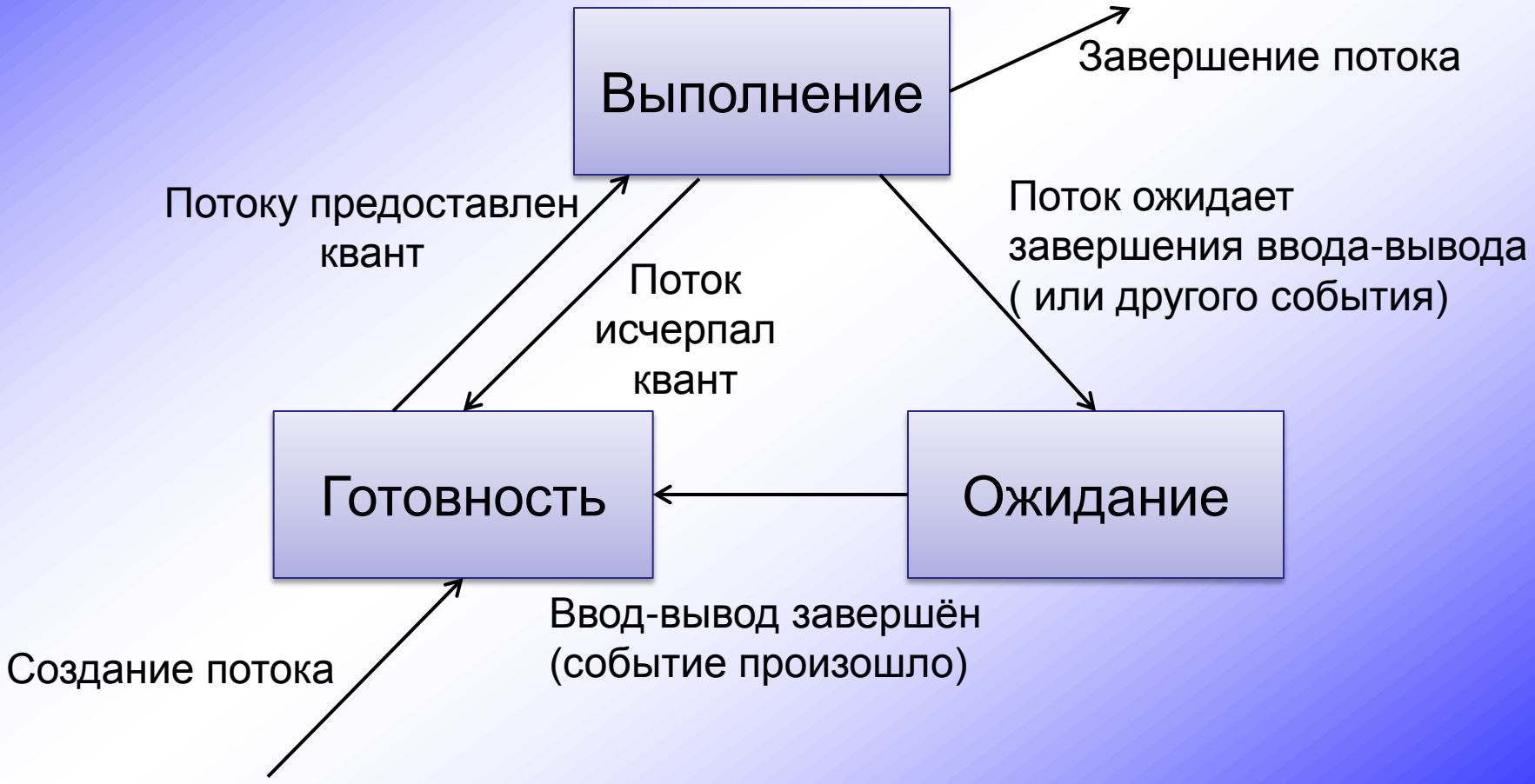
Дисциплина планирования – правило формирования очереди готовых к выполнению задач:

- алгоритмы, основанные на квантовании;
- алгоритмы, основанные на приоритетах.

Алгоритмы планирования, основанные на квантовании

- Квант – выдаваемый потоку ограниченный непрерывный период процессорного времени.
- При исчерпании кванта поток переносится в конец очереди.
- Все алгоритмы не используют априорные знания о потоке, а используют только информации о работе потока в системе.

Граф состояний потока в системе с квантованием



Причины смены активного потока

- Поток завершился и покинул систему.
- Произошла ошибка.
- Поток перешёл в состояние ожидания.
- Поток исчерпал выделенный квант процессорного времени.

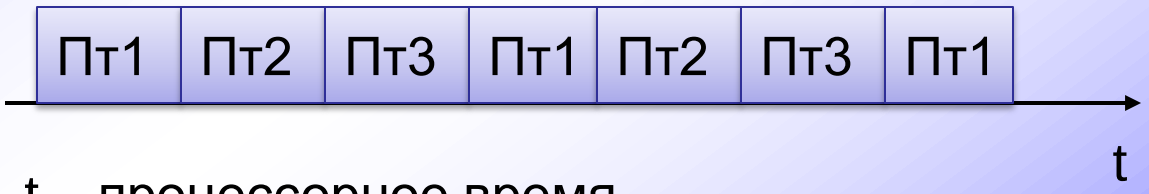
Алгоритмы планирования, основанные на квантовании

- Одинаковая длительность квантов.
- Уменьшение длительности кванта после первого цикла выполнения потока.
- Увеличение длительности кванта после первого цикла выполнения потока.
- Предпочтение потоков, обращающихся к устройствам ввода-вывода.

Алгоритм планирования с одинаковой длительностью кванта

При увеличении длительности кванта:

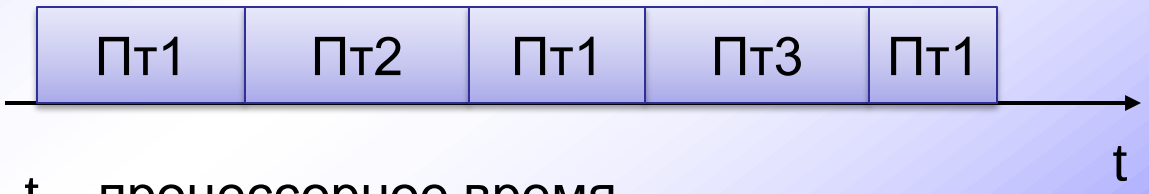
- увеличивается вероятность завершения работы потока за один цикл;
- приближение работы системы к однозадачной.



t – процессорное время.
Пт – поток.

Алгоритм планирования с уменьшением кванта

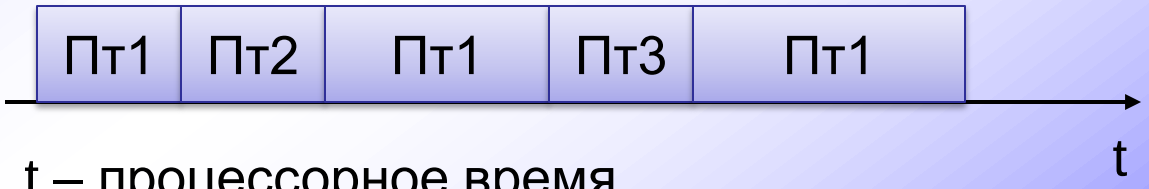
- При невыполнении потока за один цикл длительность каждого следующего кванта уменьшается до определённого значения.
- Быстрое выполнение небольших задач.
- Переход длительных задач в фоновый режим.



t – процессорное время.
Пт – поток.

Алгоритм планирования с увеличением кванта

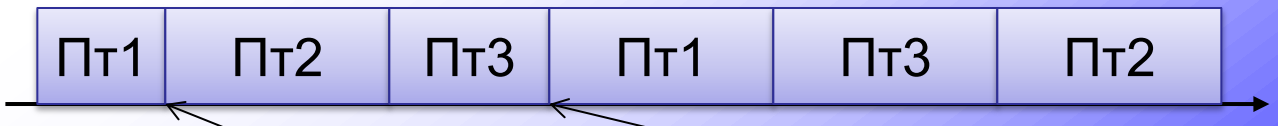
- При невыполнении потока за один цикл длительность каждого следующего кванта увеличивается до определённого значения.
- Используется в системах с большим количеством задач, требующих длительных вычислений.
- Снижаются расходы процессорного времени на переключения с потока на поток.



t – процессорное время.
Пт – поток.

Алгоритм планирования с предпочтением потоков обращающихся к вводу-выводу

- Потоки часто обращающиеся к вводу-выводу используют квант процессорного времени не полностью.
- Отдельная очередь потоков, в предыдущем цикле не полностью использовавших квант, просматривается в первую очередь.

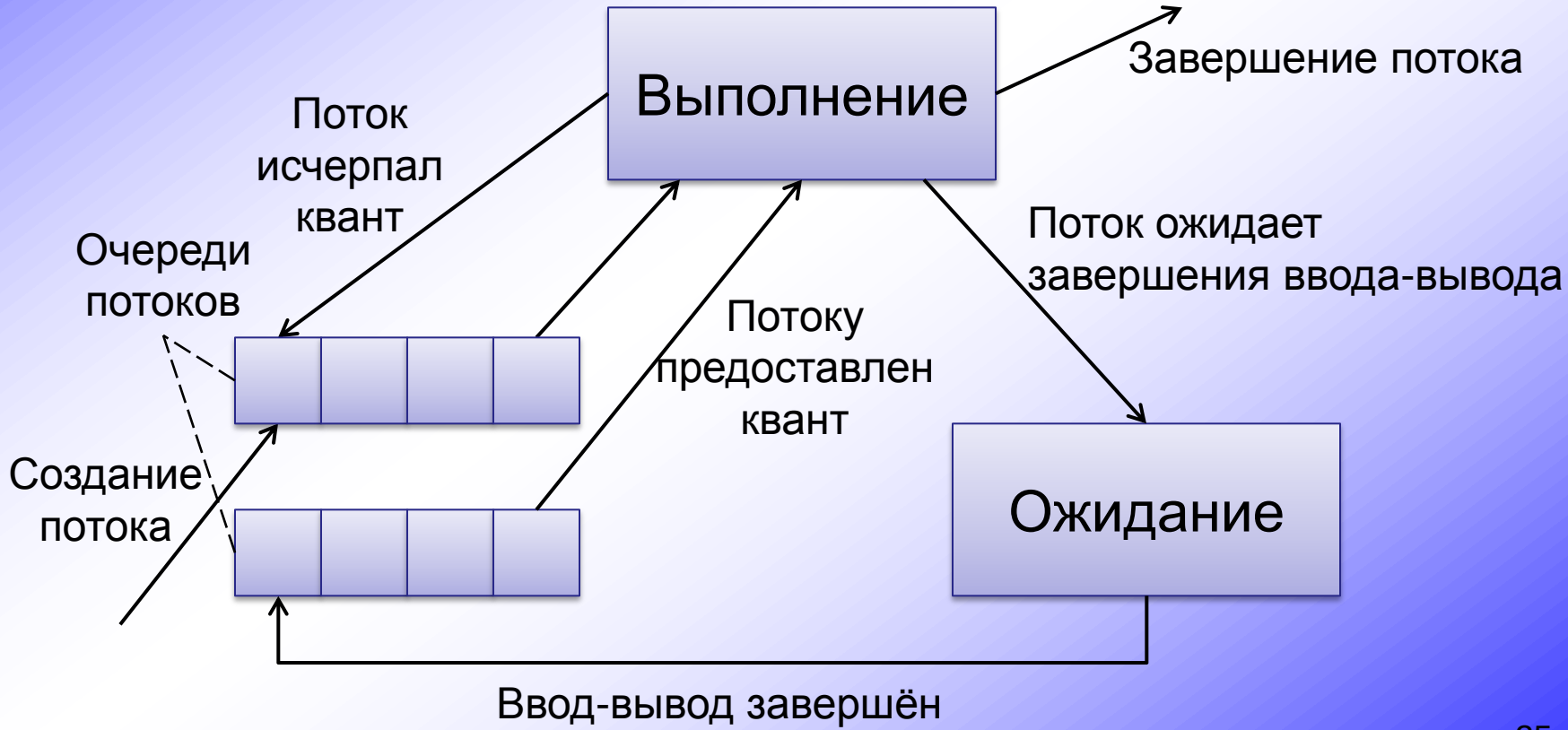


t – процессорное время.
Пт – поток.

Обращение к устройству ввода-вывода потока1

Завершение операции ввода-вывода для потока1

Граф состояний потока при предпочтении потоков, обращающихся к вводу-выводу



Алгоритмы планирования, основанные на приоритетах

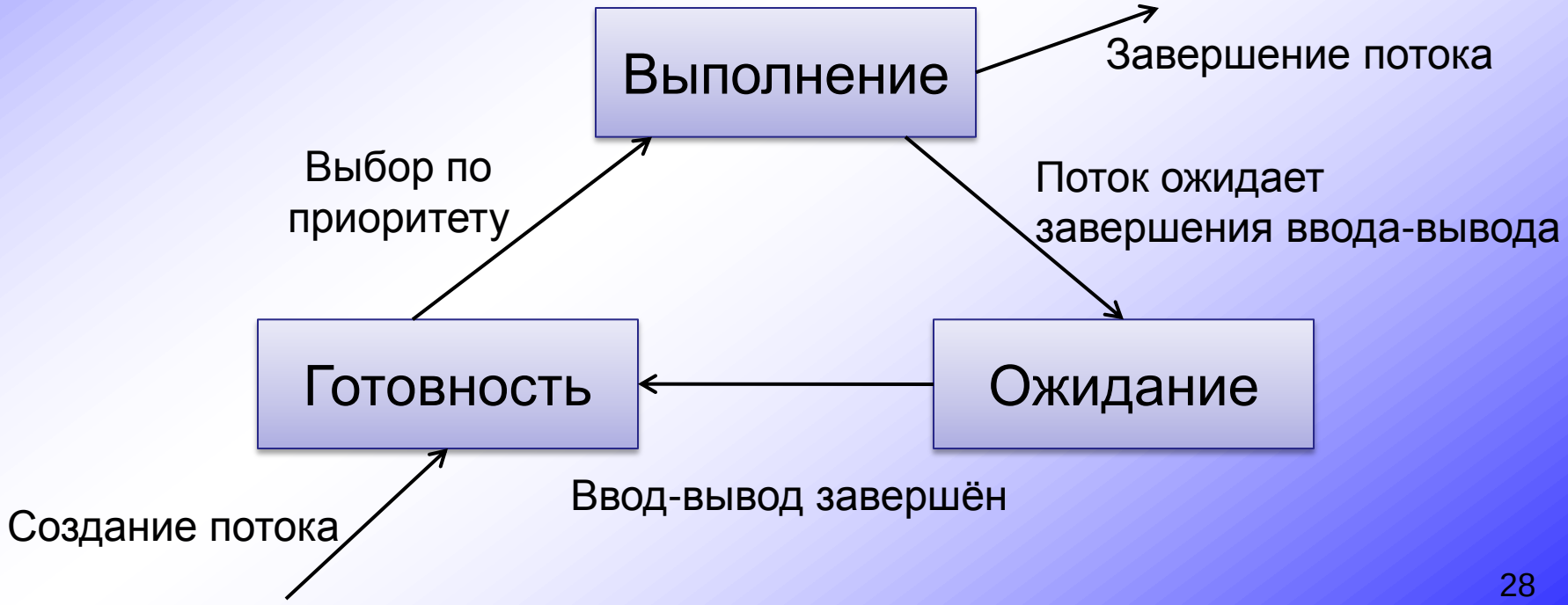
- Приоритет – число, присваиваемое потоку, которое характеризует его степень привилегированности при использовании ресурсов системы.
- При назначении приоритета ОС учитывает приоритет создателя данного потока (пользователя, другого потока).

Алгоритмы планирования, основанные на приоритетах

- ОС поддерживает свою очередь потоков для каждого значения приоритета. Потоки, перешедшие в состояние «готовность», помещаются в конец очереди, соответствующей приоритету потока.
- При использовании статических (неизменяемых) приоритетов существует возможность невыполнения низкоприоритетных задач.

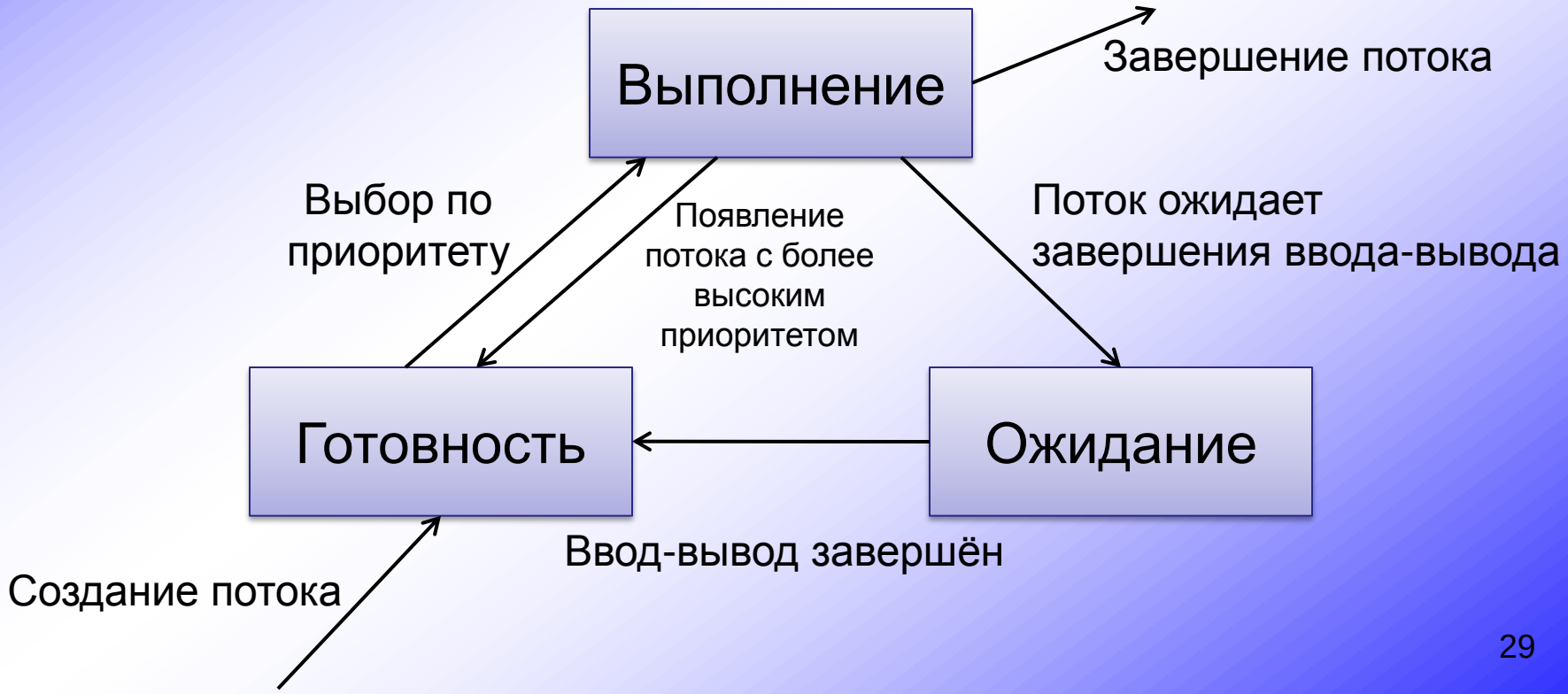
Планирование с относительными приоритетами

При появлении в очереди потока с более высоким приоритетом активный поток выполняется до тех пор пока сам не покинет процессор.



Планирование с абсолютными приоритетами

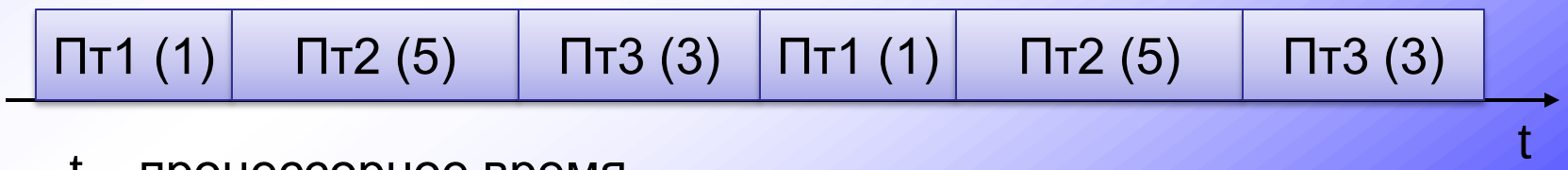
При появлении в очереди потока с более высоким приоритетом активный поток сразу завершает свою работу.



Смешанные алгоритмы планирования

Основаны на квантовании, но величина кванта и/или порядок выбора потока из очереди определяются приоритетами:

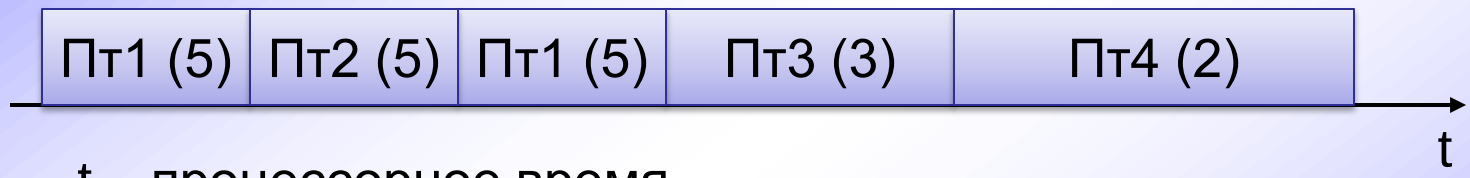
- выбор из очереди каждого следующего потока, длительность кванта больше для потоков с более высоким приоритетом;



t – процессорное время.
Пт – поток (в скобках – приоритет потока).

Смешанные алгоритмы планирования

- выбор из очереди по приоритетам, длительность кванта больше для низкого приоритета;



t – процессорное время.
Пт – поток (в скобках – приоритет потока).

- возможно разделение потоков на приоритетные группы, в которых также происходит градация по приоритетам.

Динамические приоритеты

- Динамические приоритеты могут изменяться во время существования потока.
- Изначально потоку присваивается базовый уровень приоритета – уровень приоритета, получаемый комбинацией относительного приоритета потока и класса приоритета процесса, которому принадлежит данный поток.
- Изменение динамического приоритета может происходить по инициативе пользователя, самого потока или ОС.

Приоритеты процессов в Windows NT

Класс приоритета	Описание
Real-lime	Потоки в этом процессе обязаны немедленно реагировать на события, обеспечивая выполнение критических по времени задач. Такие потоки вытесняют даже компоненты операционной системы.
High	Потоки в этом процессе тоже должны немедленно реагировать на события, обеспечивая выполнение критических по времени задач. Этот класс присвоен, например, Диспетчеру задач, что даёт возможность пользователю закрывать больше неконтролируемые процессы
Above normal	Класс приоритета, промежуточный между normal и high.
Normal	Потоки в этом процессе не предъявляют особых требований к выделению им процессорного времени.
Below normal	Класс приоритета, промежуточный между normal и idle.
Idle	Потоки в этом процессе выполняются, когда система не занята другой работой. Этот класс приоритета обычно используется для утилит, работающих в фоновом режиме, экранных заставок и приложений, собирающих статистическую информацию.

Относительные приоритеты потоков в Windows NT

Относительный приоритет потока	Описание
Time-critical	Поток выполняется с приоритетом 31 в классе real-time и с приоритетом 15 в других классах.
Highest	Поток выполняется с приоритетом на два уровня выше обычного для данного класса.
Above normal	Поток выполняется с приоритетом на один уровень выше обычного для данного класса.
Normal	Поток выполняется с обычным приоритетом процесса для данного класса.
Below normal	Поток выполняется с приоритетом на один уровень ниже обычного для данного класса.
Lowest	Поток выполняется с приоритетом на два уровня ниже обычного для данного класса.
Idle	Поток выполняется с приоритетом 16 в классе real-time и с приоритетом 1 в других классах.

Значения приоритета потоков в Windows NT

Относительный приоритет потока	Класс приоритета процесса					
	Idle	Below normal	Normal	Above normal	High	Real-time
Time-critical (критичный по времени)	15	15	15	15	15	31
Highest (высший)	6	8	10	12	15	26
Above normal (выше обычного)	5	7	9	11	14	25
Normal (обычный)	4	6	8	10	13	24
Below normal (ниже обычного)	3	5	7	9	12	23
Lowest (низший)	2	4	6	8	11	22
Idle (простаивающий)	1	1	1	1	1	16

Принципы формирования динамического приоритета

- Потоки, выполняющиеся в режиме ядра имеют более высокий приоритет.
- ОС может дифференцированно повышать приоритет в зависимости от типа события, не позволившего использовать квант полностью (большее повышение при ожидании ввода данных, меньшее – при дисковых операциях).

Принципы формирования динамического приоритета

- ОС может изменять приоритет в зависимости от полноты использования кванта процессорного времени (понижать в случае полного использования и наоборот).
- ОС разделения времени может повышать приоритет активной задачи (с окном которой работает пользователь) для обеспечения её быстрого отклика.
- ОС может повышать приоритет задачи, долгое время не выбиравшейся на выполнение.

Задачи алгоритмов планирования

Системы пакетной обработки:

- обеспечение наиболее быстрого выполнения вычислений;
- минимизация количества переключения потоков.

Системы разделения времени:

- обеспечение наиболее быстрого выполнения операций ввода-вывода;
- обеспечение ощущения непрерывной работы с приложениями у пользователя.

Системы реального времени:

- обеспечение выполнения всех критических операций в срок.

Синхронизация процессов и потоков

Межпроцессное взаимодействие

Возникающие проблемы:

- передача информации от одного процесса к другому (только для процессов);
- предъявление различными потоками требований на одни и те же ресурсы;
- согласование действий потоков в случае, если выходные данные одного потока являются входными данными другого.

Синхронизация потоков

Синхронизация потоков – согласование скоростей параллельно выполняющихся потоков путём приостановки потока до наступления какого-либо события и последующей его активизации при наступлении этого события.

Задачи, решаемые с помощью синхронизации

- Обработка процессом некоторых аппаратных прерываний.
- Терминальное управление процессами.
- Синхронизация параллельных процессов, выполняющих действия с общей областью оперативной памяти.
- Синхронизация параллельных процессов, выполняющих информационный обмен при использовании общей области оперативной памяти.

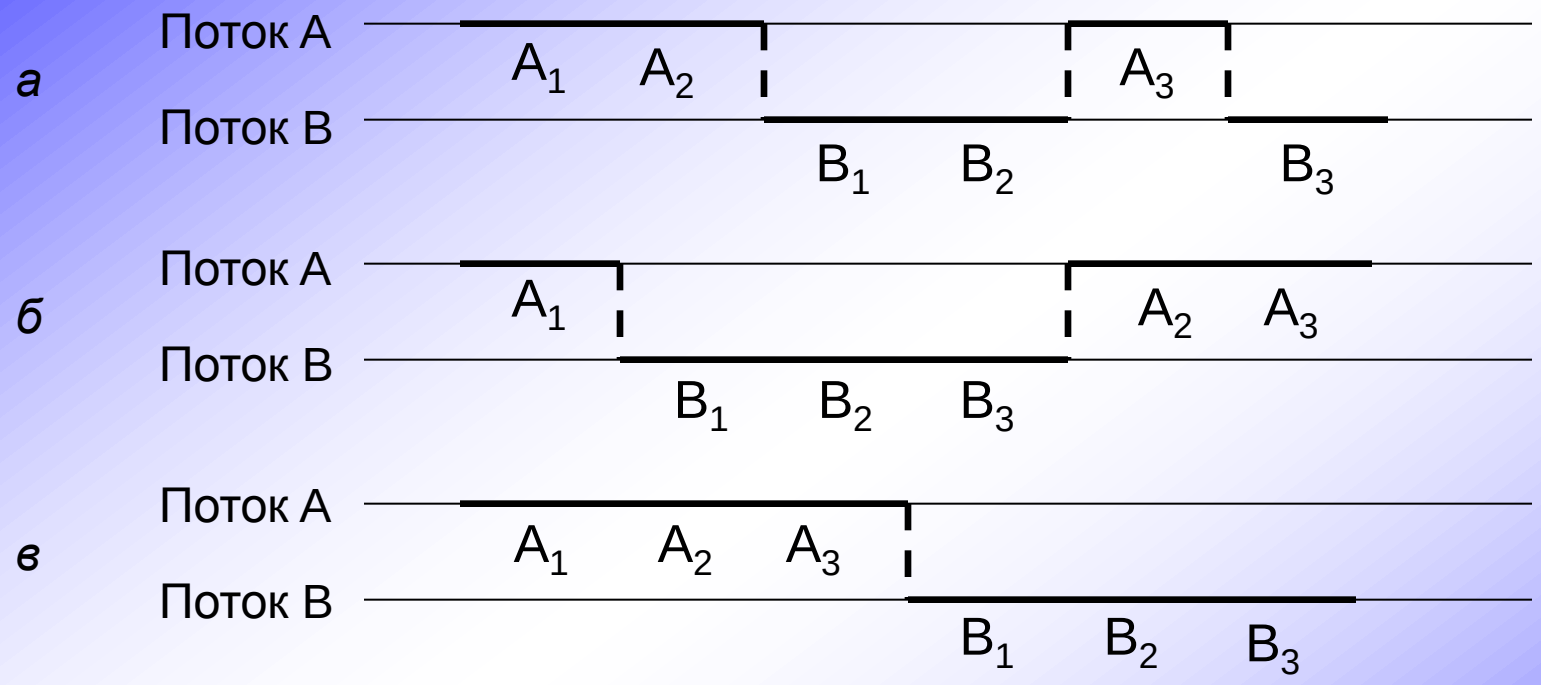
Состояние состязания (гонка)

Состояние состязания – ситуация, когда два или более потоков работают с разделяемыми ресурсами (устройствами или данными) и конечный результат зависит от соотношения скоростей выполнения потоков.

Состояние состязания

1. Считывание из базы данных в буфер записи о клиенте (A_1 – при изменении поля «Заказ», B_1 – при изменении поля «Оплата»).
2. Внесение нового значения в поле «Заказ» (A_2) или «Оплата» (B_2).
3. Внесение изменённой записи в базу данных (A_3 – при изменении поля «Заказ», B_3 – при изменении поля «Оплата»).

Состояние состязания



- *а* – потеря данных о заказе;
- *б* – потеря данных об оплате;
- *в* – корректное изменение записей;
- пунктир – смена активного потока.

Критические области

Критическая область – часть кода программы, в которой есть обращение к ресурсам, используемым текущим потоком совместно с другими потоками, что может привести к непредсказуемым последствиям.

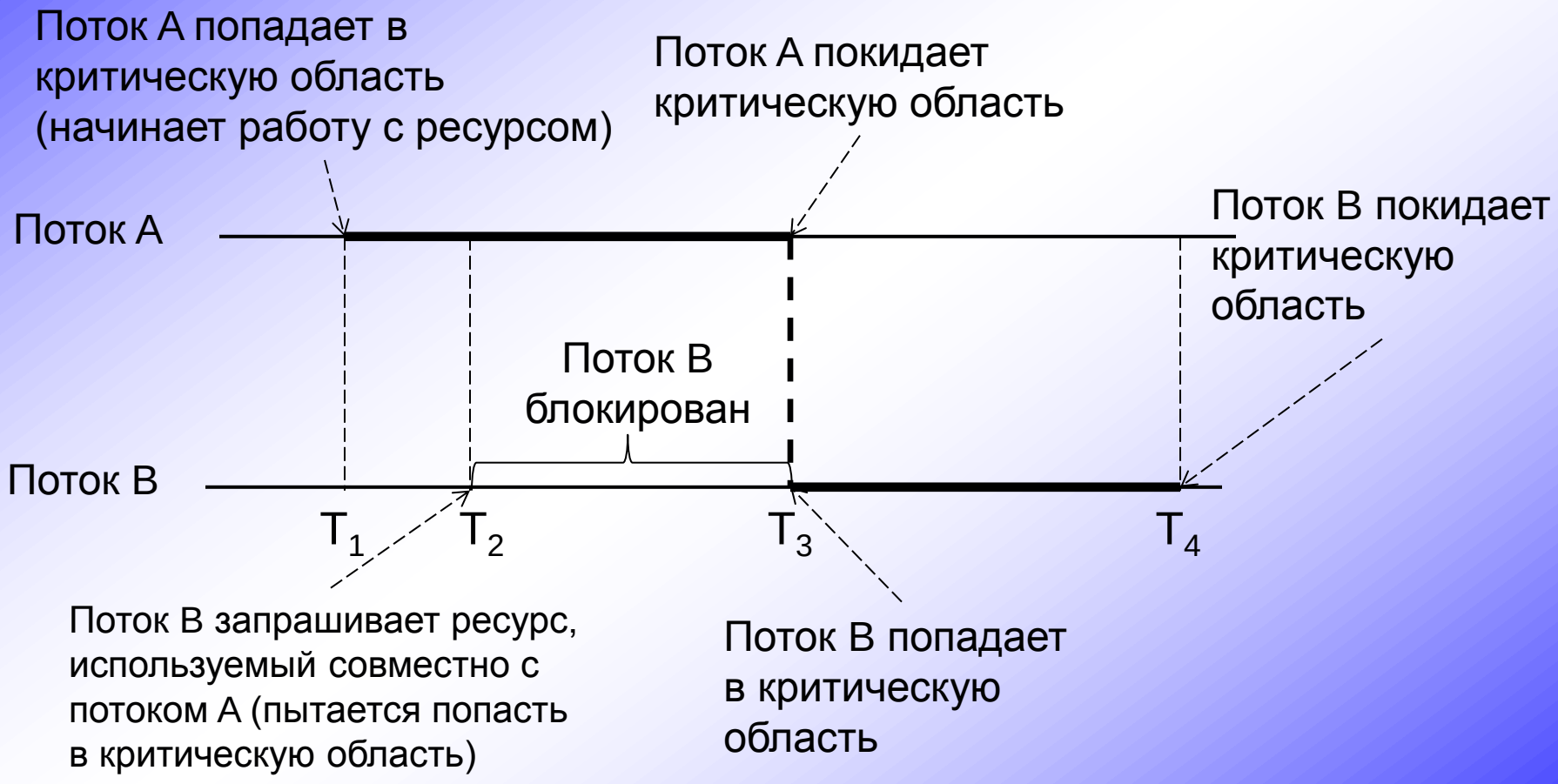
Подобные ресурсы называются критическими.

Взаимное исключение

Взаимное исключение – запрет для всех потоков на использование совместно используемых ресурсов во время их использования текущим потоком, вне зависимости от его состояния.

Таким образом, в критической области в каждый момент времени может находиться только один поток.

Взаимное исключение с использованием критических областей



Условия правильной совместной работы параллельных потоков

- Два потока не должны одновременно находиться в критических областях.
- В программе не должно быть предположений о скорости или количестве процессоров.
- Поток, находящийся вне критической области, не должен блокировать другие потоки.
- Невозможность ситуации, в которой поток вечно ждёт попадания в критическую область.

Взаимное исключение с активным ожиданием

- Запрет всех прерываний при входе потока в критическую область и разрешение при выходе.
- Блокирующие переменные.
- Команда TSL.

Запрет всех прерываний

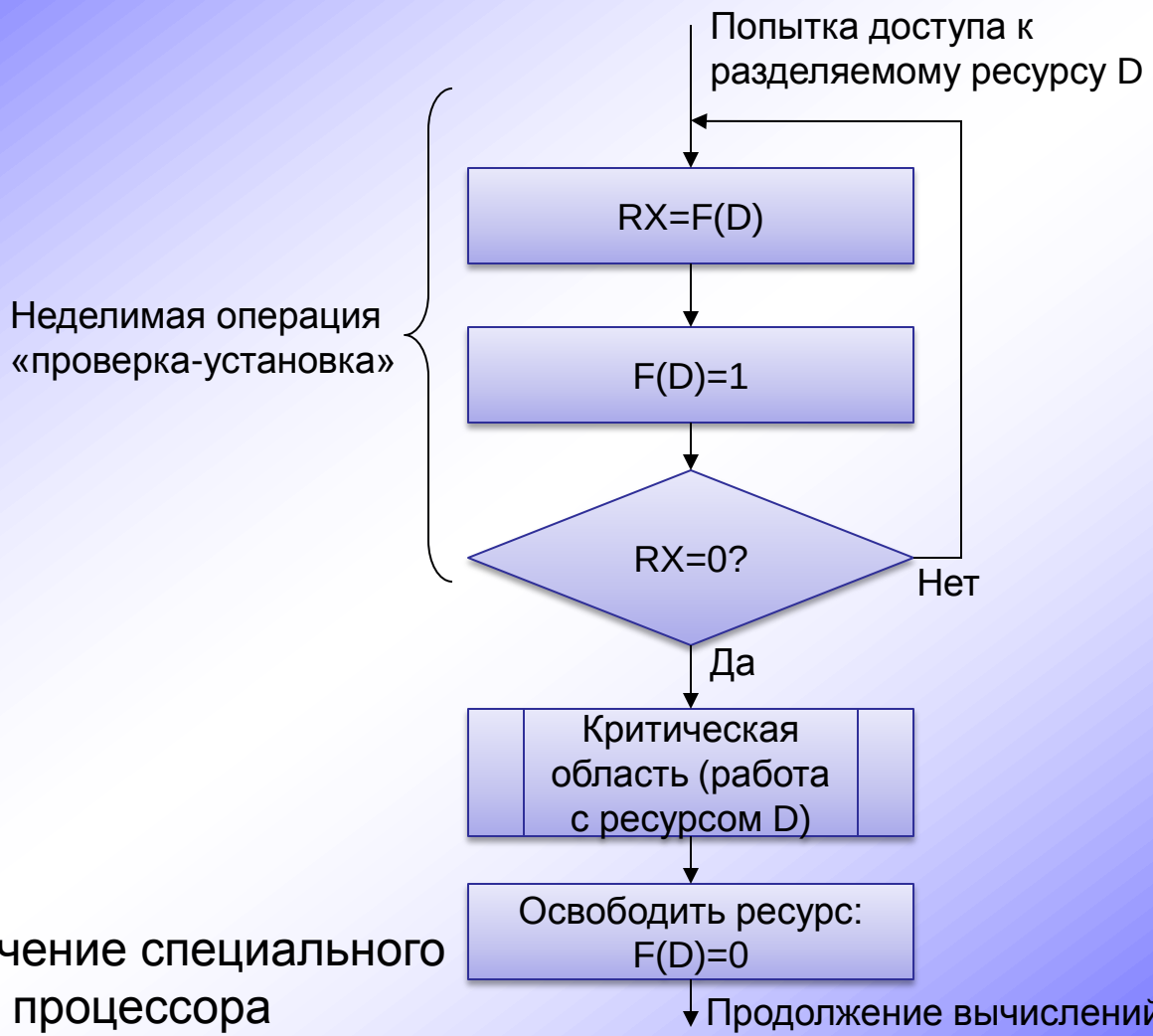
- Активный поток не может смениться, так как запрещены все прерывания, включая прерывание по таймеру.
- Используется в ОС, например, в моменты перепланировки.
- Недостатки: пользовательский поток может надолго занять процессор; возможна некорректная работа в мультипроцессорных системах.

Блокирующие переменные

F(D) – блокирующая переменная для ресурса D.

Команда TSL

TSL (Test and Set Lock – проверить и заблокировать)



RX – значение специального регистра процессора

Недостатки алгоритмов взаимного исключения с активным ожиданием

- В течение времени, когда один поток находится в критической области, другой поток, требующий тот же ресурс, при предоставлении ему процессорного времени будет выполнять цикл опроса занятости ресурса.
- При вытесняющем приоритетном планировании поток с более низким приоритетом может занять критическую область, а поток с более высоким приоритетом вытеснит его и будет бесконечно выполнять цикл опроса.

Поток как объект синхронизации

Потоки при помощи системных вызовов могут синхронизировать своё выполнение с состоянием некоторого объекта.

Например в зависимости от поставленных условий:

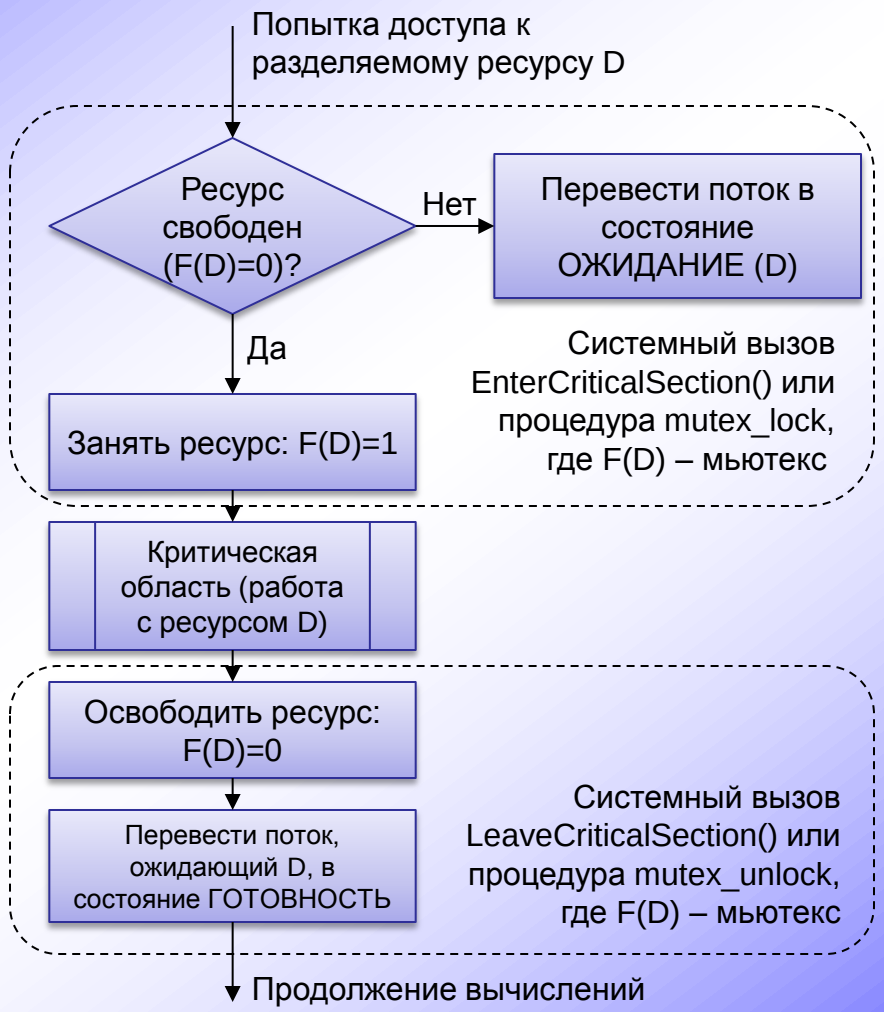
- возможен перевод потока X в состояние «ожидание» при помощи системного вызова sleep(X), пока его не запустит другой поток;
- возможен перевод потока X в состояние «готовность» при помощи системного вызова wakeup(X).

Выполнение потока может быть синхронизовано с одним или несколькими объектами.

Мьютекс

Мьютекс – переменная, которая может находиться в одном из двух состояний: блокированном или неблокированном (двоичный семафор).

Мьютексы и системные вызовы для работы с критическими областями



Семафоры

Семафор – переменная, которая может содержать целые неотрицательные значения и используется для синхронизации вычислительных процессов.

Операции для работы с семафором S:

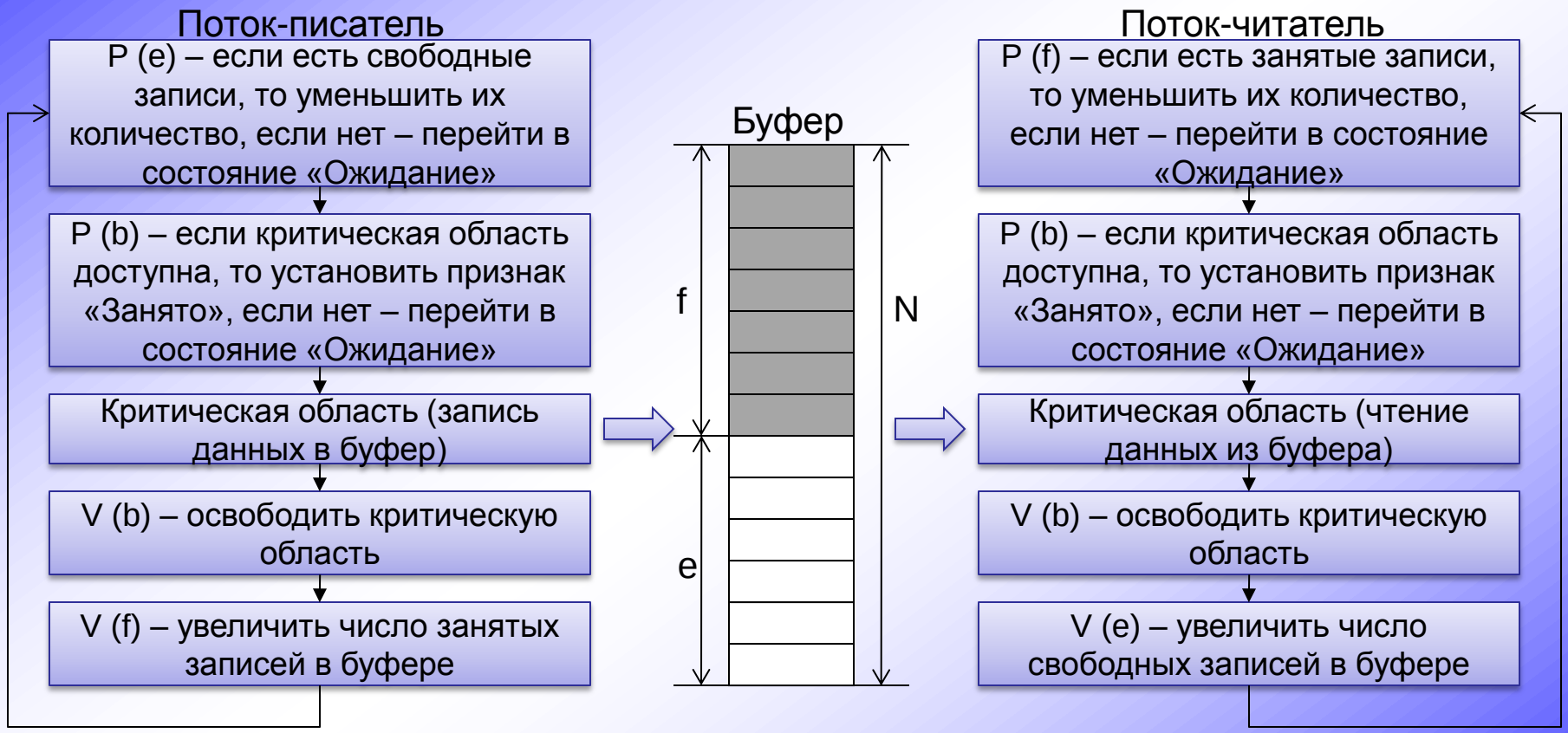
- V(S) – переменная S увеличивается на 1
- P(S) – переменная S уменьшается на 1, если это возможно. Если S=0, то поток, вызывающий операцию P, переходит в состояние «ожидание» и ждёт пока уменьшение станет возможным.

Обе операции являются элементарными действиями, т.е. едины и неделимы.

Проблема производителя и потребителя

- Два потока одновременно используют буфер ограниченного размера.
- Один поток помещает данные в буфер, а другой – считывает данные.
- Необходимо обеспечить корректную работу потока-писателя, если буфер полон, и потока-читателя, если буфер пуст.

Пример работы семафоров. Проблема производителя и потребителя



Семафоры:

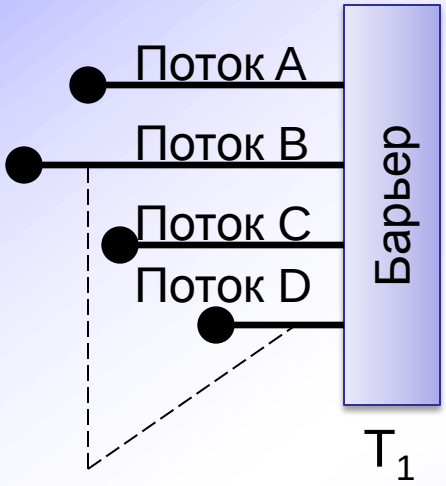
e – количество пустых записей в буфере (начальное значение - N);

f – количество заполненных записей в буфер (начальное значение - 0);

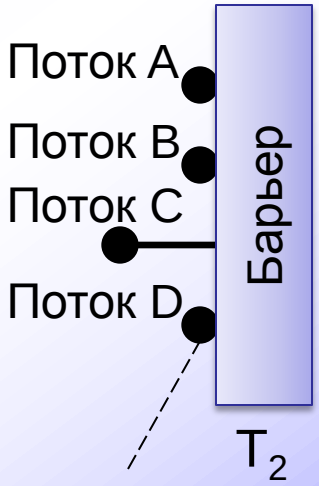
b – двоичный семафор.

Барьеры

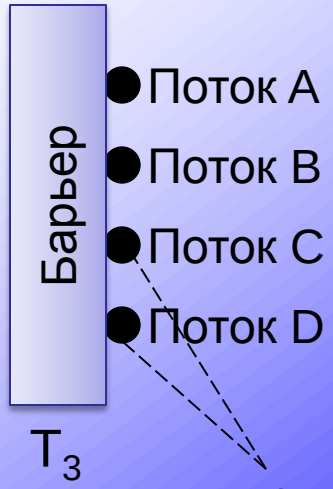
Барьер – механизм синхронизации для групп потоков с несколькими фазами выполнения. В конце каждой фазы ставится барьер. Для прохождения барьера, его должны достигнуть все потоки в группе.



Процессорное время, необходимое потоку для достижения барьера



Выполнение потока достигло барьер



Все потоки достигли барьер и продолжают выполнение

Рассмотренные вопросы

- Состояния потоков.
- Стратегии и дисциплины планирования.
- Синхронизация процессов и потоков.
- Методы и средства синхронизации.

Всем спасибо – все свободны,

если нет вопросов