Лекция №7

Планирование процессов и потоков

Одной из основных подсистем мультипрограммной ОС является подсистема управления процессами и потоками, которая занимается их созданием и уничтожением, поддерживает взаимодействие между ними, а также распределяет процессорное время между несколькими одновременно существующими в системе процессами и потоками.

Подсистема управления процессами и потоками ответственна за обеспечение процессов необходимыми ресурсами. ОС поддерживает в памяти специальные информационные структуры, в которые записывает, какие ресурсы выделены каждому процессу. Некоторые из ресурсов выделяются процессу при его создании, а некоторые — динамически по запросам во время выполнения. При выполнении этих функций подсистема управления процессами взаимодействует с другими подсистемами ОС, ответственными за управление ресурсами, такими как подсистема управления памятью, подсистема ввода-вывода, файловая система.

Хотя потоки возникают и выполняются асинхронно, у них может возникнуть необходимость во взаимодействии, например при обмене данными. Также очень важно и согласование скоростей потоков.

Каждый раз, когда процесс завершается, ОС предпринимает шаги, чтобы «зачистить следы» его пребывания в системе. Подсистема управления процессами закрывает все файлы, с которыми работал процесс, освобождает области оперативной памяти, отведенные под коды, данные и системные информационные структуры процесса. Выполняется коррекция всевозможных очередей ОС и списков ресурсов, в которых имелись ссылки на завершаемый процесс.

Понятия «процесс» и «поток»

Чтобы поддерживать мультипрограммирование, ОС должна определить и оформить для себя те внутренние единицы работы, между которыми будет разделяться процессор и другие ресурсы компьютера. В настоящее время в большинстве операционных систем определены два типа единиц работы. Более крупная единица работы, обычно носящая название ***процесса***, или задачи, требует для своего выполнения нескольких более мелких работ, для обозначения которых используют термины «***поток***», или «***нить***».

При использовании этих терминов часто возникают сложности в силу нескольких причин. Во-первых, — специфика различных ОС, когда совпадающие по сути понятия получили разные названия. Во-вторых, по мере развития системного программирования и методов организации вычислений некоторые из этих терминов получили новое смысловое значение, особенно это касается понятия «процесс», который уступил многие свои свойства новому понятию «поток». В-третьих, терминологические сложности порождаются наличием нескольких вариантов перевода англоязычных терминов на русский язык. Например, термин «thread» переводится как «нить», «поток», «облегченный процесс», «минизадача» и др. Далее в качестве названия единиц работы ОС будут использоваться термины «процесс» и «поток». В тех же случаях, когда различия между этими понятиями не будут играть существенной роли, они объединяются под обобщенным термином «задача».

В чем отличия в понятиях «процесс» и «поток»?

И с процессом, и с потоком связывается определенный программный код, который оформляется в виде исполняемого модуля. Чтобы этот программный код мог быть выполнен, его необходимо загрузить в оперативную память, возможно, выделить некоторое место на диске для хранения данных, предоставить доступ к устройствам ввода-вывода. В ходе выполнения программе может также понадобиться доступ к информационным ресурсам.

В операционных системах, где существуют и процессы, и потоки, процесс рассматривается операционной системой как заявка на потребление всех видов ресурсов, кроме одного — процессорного времени. Этот последний важнейший ресурс распределяется операционной системой между другими единицами работы — потоками, которые представляют собой последовательности (потоки выполнения) команд.

В простейшем случае процесс состоит из одного потока. В таких системах понятие «поток» полностью поглощается понятием «процесс». ОС обеспечивает каждый процесс отдельным виртуальным адресным пространством, так что ни один процесс не может получить прямого доступа к командам и данным другого процесса.

При необходимости взаимодействия процессы обращаются к операционной системе, которая, выполняя функции посредника, предоставляет им средства межпроцессной связи — конвейеры, почтовые ящики, разделяемые секции памяти и некоторые другие.

Однако в системах, в которых отсутствует понятие потока, возникают проблемы при организации параллельных вычислений в рамках процесса.

Создание потоков требует от ОС меньших накладных расходов, чем процессов. В отличие от процессов, которые принадлежат разным все потоки одного процесса всегда принадлежат одному приложению, поэтому ОС изолирует потоки в гораздо меньшей степени, нежели процессы в традиционной мультипрограммной системе. Все потоки одного процесса используют общие файлы, таймеры, устройства, одну и ту же область оперативной памяти, одно и то же адресное пространство. Поскольку каждый поток может иметь доступ к любому виртуальному адресу процесса, один поток может использовать стек другого потока. Между потоками одного процесса нет полной защиты, потому что, во-первых, это невозможно, а во-вторых, не нужно. Чтобы организовать взаимодействие и обмен данными, потокам вовсе не требуется обращаться к ОС, им достаточно использовать общую память — один поток записывает данные, а другой читает их. С другой стороны, потоки разных процессов по-прежнему хорошо защищены друг от друга.

Итак, мультипрограммирование более эффективно на уровне потоков, а не процессов.

Наибольший эффект от введения многопоточной обработки достигается в мультипроцессорных системах, в которых потоки, в том числе и принадлежащие одному процессу, могут выполняться на разных процессорах действительно параллельно (а не псевдопараллельно).

Создание процессов и потоков

Создать процесс — это прежде всего означает создать описатель процесса, т. е. одну или несколько информационных структур, содержащих все сведения о процессе, необходимые операционной системе для управления им.

Создание процесса включает загрузку кодов и данных исполняемой программы данного процесса с диска в оперативную память. Затем необходимо считать программу в выделенные для нее участки памяти и, возможно, изменить параметры программы в зависимости от размещения в памяти. В системах с виртуальной памятью в начальный момент может загружаться только часть кодов и данных процесса, с тем чтобы «подкачивать» остальные по мере необходимости. Существуют системы, в которых на этапе создания процесса не требуется непременно загружать коды и данные в оперативную память, вместо этого исполняемый модуль копируется из того каталога файловой системы, в котором он изначально находился, в область подкачки — специальную область диска, отведенную для хранения кодов и данных процессов. При выполнении всех этих действий подсистема управления процессами тесно взаимодействует с подсистемой управления памятью и файловой системой.

В многопоточной системе при создании процесса ОС создает для каждого процесса как минимум один поток выполнения. При создании потока так же, как и при создании процесса, операционная система генерирует специальную информационную структуру — описатель потока, который содержит идентификатор потока, данные о правах доступа и приоритете, о состоянии потока и другую информацию.

Планирование и диспетчеризация потоков

На протяжении существования процесса выполнение его потоков может быть многократно прервано и продолжено.

Переход от выполнения одного потока к другому осуществляется в результате планирования и диспетчеризации. Работа по определению того, в какой момент необходимо прервать выполнение текущего активного потока и какому потоку предоставить возможность выполняться, называется планированием. ОС планирует выполнение потоков независимо от того, принадлежат ли они одному или разным процессам.

Планирование потоков, по существу, включает в себя решение двух задач:

- определение момента времени для смены текущего активного потока;
- выбор для выполнения потока из очереди готовых потоков.

Существует множество различных алгоритмов планирования потоков, по-своему решающих каждую из приведенных выше задач. Именно особенности реализации планирования потоков в наибольшей степени определяют специфику операционной системы, в частности, является ли она системой пакетной обработки, системой разделения времени или системой реального времени.

Диспетчеризация заключается в реализации найденного в результате планирования решения, то есть в переключении процессора с одного потока на другой. Прежде чем прервать выполнение потока, ОС запоминает его контекст, с тем чтобы впоследствии использовать эту информацию для последующего возобновления выполнения данного потока. Контекст отражает, во-первых, состояние аппаратуры компьютера в момент прерывания потока: значение счетчика команд, содержимое регистров общего назначения, режим работы процессора, флаги, маски прерываний и другие параметры. Во-вторых, контекст включает параметры операционной среды, а именно ссылки на открытые файлы, данные о незавершенных операциях ввода-вывода, коды ошибок выполняемых данным потоком системных вызовов и т. д.

Диспетчеризация сводится к следующему:

- сохранение контекста текущего потока, который требуется сменить;
- загрузка контекста нового потока, выбранного в результате планирования;
- запуск нового потока на выполнение.

Состояния потока

ОС выполняет планирование потоков, принимая во внимание их состояние. В мультипрограммной системе поток может находиться в одном из трех основных состояний:

- выполнение — активное состояние потока, во время которого поток обладает всеми необходимыми ресурсами и непосредственно выполняется процессором;
- ожидание — пассивное состояние потока, находясь в котором, поток заблокирован по своим внутренним причинам (ждет осуществления некоторого события, например завершения операции ввода-вывода, получения сообщения от другого потока или освобождения какого-либо необходимого ему ресурса);
- готовность — также пассивное состояние потока, но в этом случае поток заблокирован в связи с внешним по отношению к нему обстоятельством (имеет все требуемые для него ресурсы, готов выполняться, однако процессор занят выполнением другого потока).

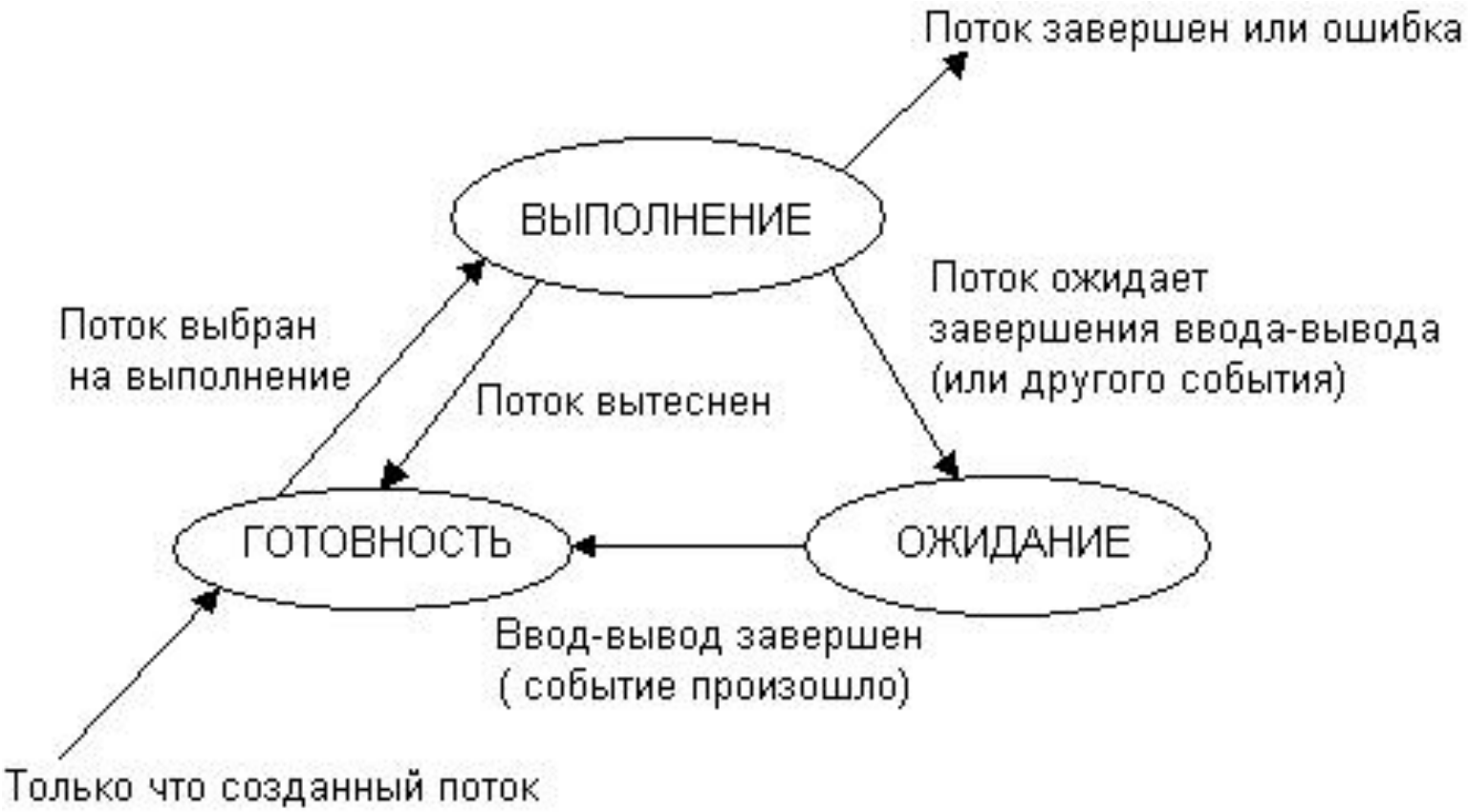

Рис. 4.3. Граф состояний потока в многозадачной среде

Только что созданный поток находится в состоянии готовности, он стоит в очереди к процессору. Когда в результате планирования подсистема управления потоками принимает решение об активизации данного потока, он переходит в состояние выполнения и находится в нем до тех пор, пока либо он сам освободит процессор, перейдя в состояние ожидания какого-нибудь события, либо будет принудительно «вытеснен» из процессора. В последнем случае поток возвращается в состояние готовности. В это же состояние поток переходит из состояния ожидания, после того как ожидаемое событие произойдет.

Рис. 4.4. Очередь потоков

В состоянии выполнения в однопроцессорной системе может находиться не более одного потока, а в каждом из состояний ожидания и готовности — несколько потоков. Эти потоки образуют очереди соответственно ожидающих и готовых потоков. Очереди потоков организуются путем объединения в списки описателей отдельных потоков. Таким образом, каждый описатель потока, кроме всего прочего, содержит по крайней мере один указатель на другой описатель, соседствующий с ним в очереди. Такая организация очередей позволяет легко их переупорядочивать, включать и исключать потоки, переводить потоки из одного состояния в другое. Если предположить, что на рис. 4.4 показана очередь готовых потоков, то запланированный порядок выполнения выглядит так: A, B, E, D, C.

Вытесняющие и невытесняющие алгоритмы планирования

С самых общих позиций все множество алгоритмов планирования можно разделить на два класса: вытесняющие и невытесняющие алгоритмы планирования.

- Невытесняющие (non-preemptive) алгоритмы основаны на том, что активному потоку позволяется выполняться, пока он сам, по собственной инициативе, не отдаст управление операционной системе для того, чтобы та выбрала из очереди другой готовый к выполнению поток.
- Вытесняющие (preemptive) алгоритмы — это такие способы планирования потоков, в которых решение о переключении процессора с выполнения одного потока на выполнение другого потока принимается операционной системой, а не активной задачей.

Основным различием между вытесняющими и невытесняющими алгоритмами является степень централизации механизма планирования потоков. При вытесняющем мультипрограммировании функции планирования потоков целиком сосредоточены в операционной системе и программист пишет свое приложение, не заботясь о том, что оно будет выполняться одновременно с другими задачами. При этом операционная система выполняет следующие функции: определяет момент снятия с выполнения активного потока, запоминает его контекст, выбирает из очереди готовых потоков следующий, запускает новый поток на выполнение, загружая его контекст.

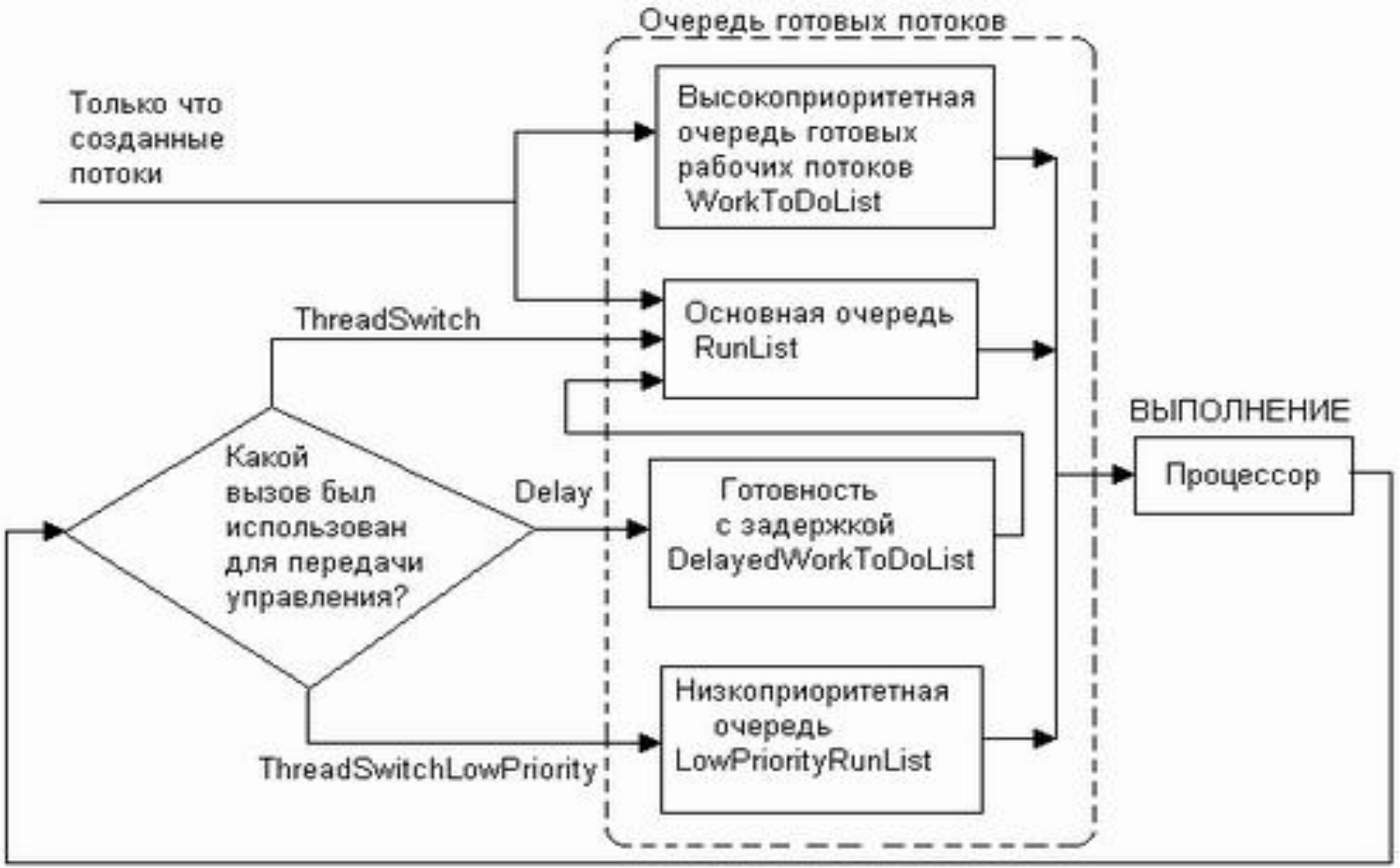

Рис. 4.5. Схема планирования потоков в NetWare

Алгоритмы планирования, основанные на квантовании

В основе многих вытесняющих алгоритмов планирования лежит концепция квантования. В соответствии с этой концепцией каждому потоку поочередно для выполнения предоставляется ограниченный непрерывный период процессорного времени — квант. Смена активного потока происходит, если:

- поток завершился и покинул систему;
- произошла ошибка;
- поток перешел в состояние ожидания;
- исчерпан квант процессорного времени, отведенный данному потоку.

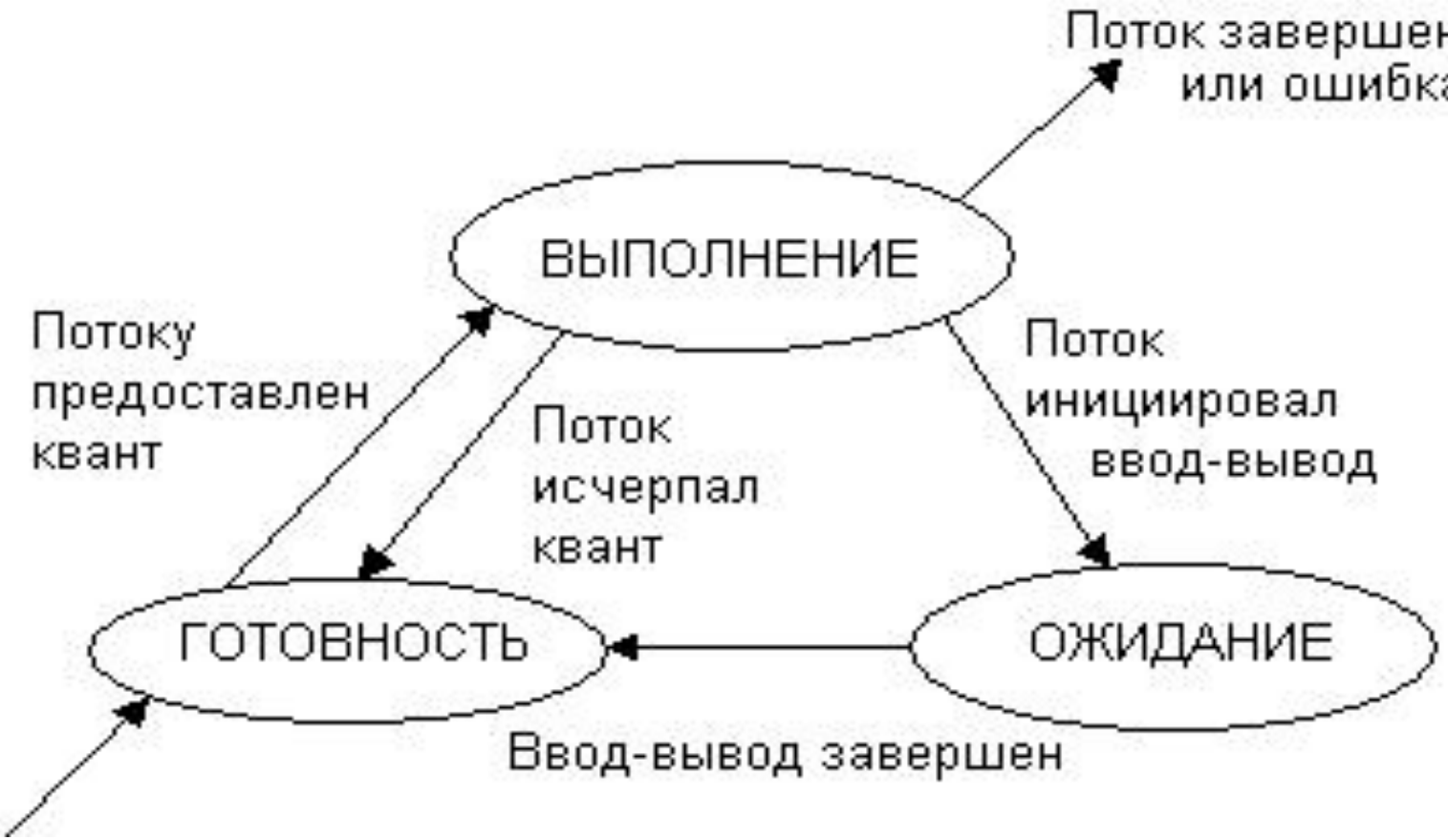

Рис. 4.6. Граф состояний потока в системе с квантованием

Поток, который исчерпал свой квант, переводится в состояние готовности и ожидает, когда ему будет предоставлен новый квант процессорного времени, а на выполнение в соответствии с определенным правилом выбирается новый поток из очереди готовых. Граф состояний потока, изображенный на рис. 4.6, соответствует алгоритму планирования, основанному на квантовании.

Рис. 4.7. Иллюстрация расчета времени ожидания в очереди

Кванты, выделяемые потокам, могут быть одинаковыми для всех потоков или различными. Рассмотрим, например, случай, когда всем потокам предоставляются кванты одинаковой длины **q** (рис. 4.7). Если в системе имеется n потоков, то время, которое поток проводит в ожидании следующего кванта, можно грубо оценить как q(n-l). Чем больше потоков в системе, тем больше время ожидания, тем меньше возможности вести одновременную интерактивную работу нескольким пользователям. Но если величина кванта выбрана очень небольшой, то значение произведения q(n-l) все равно будет достаточно мало для того, чтобы пользователь не ощущал дискомфорта от присутствия в системе других пользователей. Типичное значение кванта в системах разделения времени составляет десятки миллисекунд.

Рис. 4.8. Квантование с предпочтением потоков, интенсивно обращающихся к вводу-выводу

Потоки получают для выполнения квант времени, но некоторые из них используют его не полностью. В качестве компенсации за неиспользованные полностью кванты потоки получают привилегии при последующем обслуживании. Для этого планировщик создает две очереди готовых потоков (рис. 4.8). Очередь 1 образована потоками, которые пришли в состояние готовности в результате исчерпания кванта времени, а очередь 2 — потоками, у которых завершилась операция ввода-вывода. При выборе потока для выполнения прежде всего просматривается вторая очередь, и только если она пуста, квант выделяется потоку из первой очереди.

Алгоритмы планирования, основанные на приоритетах

Другой важной концепцией, лежащей в основе многих вытесняющих алгоритмов планирования, является приоритетное обслуживание. Приоритетное обслуживание предполагает наличие у потоков некоторой изначально известной характеристики — приоритета, на основании которой определяется порядок их выполнения. Приоритет — это число, характеризующее степень привилегированности потока при использовании ресурсов вычислительной машины, в частности процессорного времени: чем выше приоритет, тем выше привилегии, тем меньше времени будет проводить поток в очередях.

Приоритет может выражаться целым или дробным, положительным или отрицательным значением. В некоторых ОС принято, что приоритет потока тем выше, чем больше (в арифметическом смысле) число, обозначающее приоритет. В других системах, наоборот, чем меньше число, тем выше приоритет.

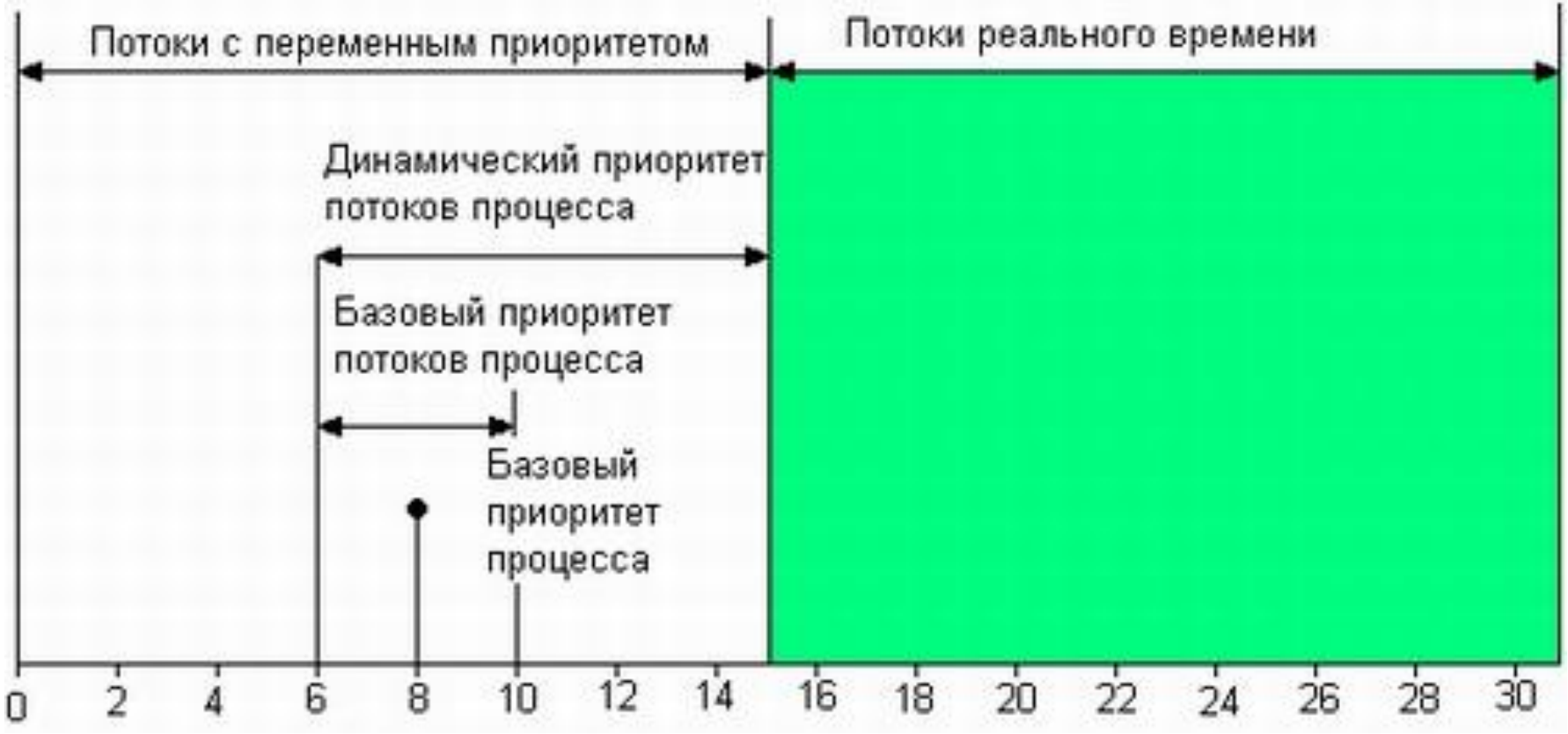

Рис. 4.9. Схема назначения приоритетов в Windows NT

Рассмотрим схему назначения приоритетов потокам, принятую в операционной системе Windows NT (рис. 4.9). В системе определено 32 уровня приоритетов и два класса потоков — потоки реального времени и потоки с переменными приоритетами. Диапазон от 1 до 15 включительно отведен для потоков с переменными приоритетами, а от 16 до 31 — для более критичных ко времени потоков реального времени (приоритет 0 зарезервирован для системных целей).

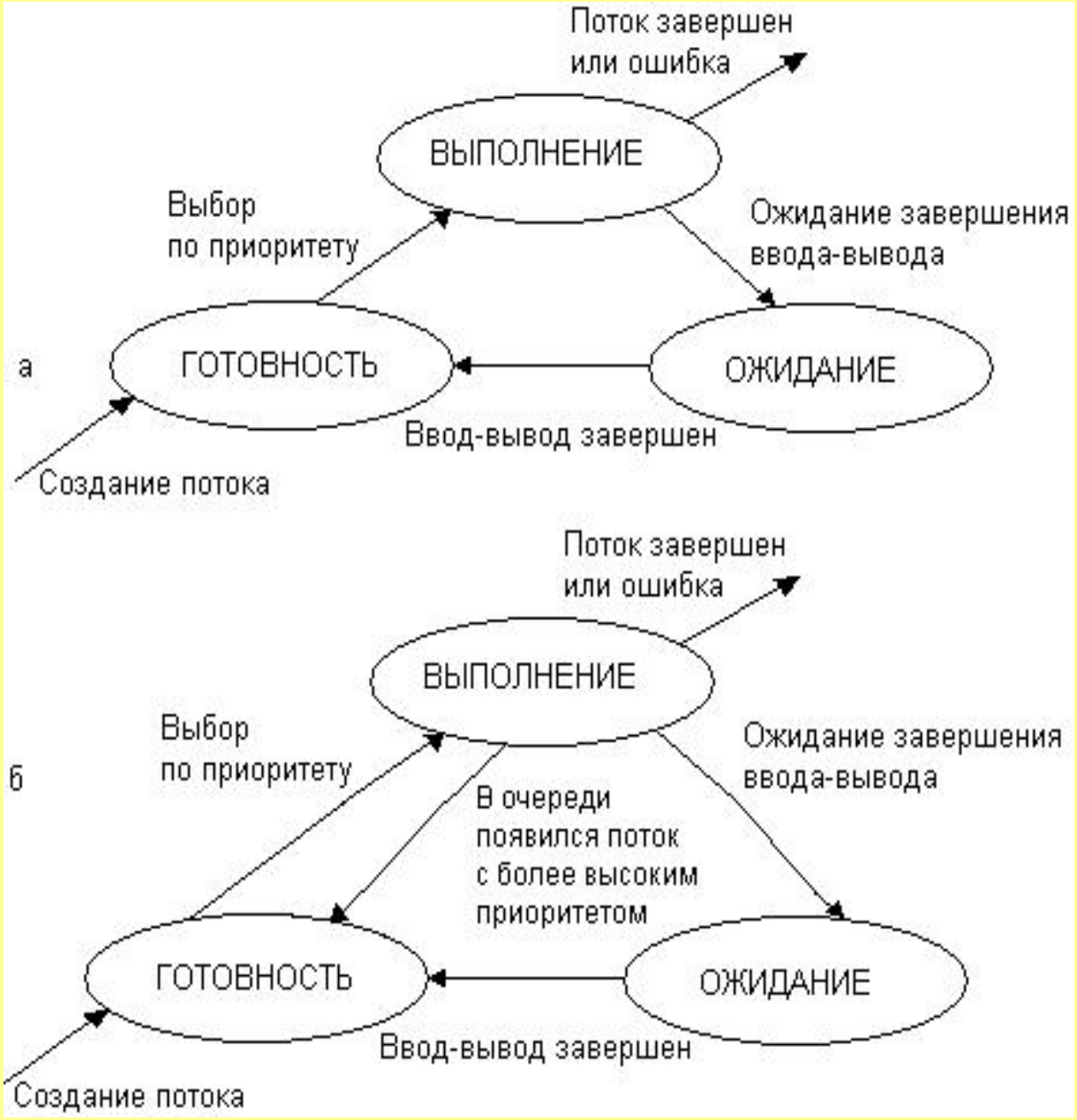

Рис. 4.10. Графы состояний потоков в системах с относительными и абсолютными приоритетами

В системах с относительными приоритетами (рис. 4.10) активный поток выполняется до тех пор, пока он сам не покинет процессор, перейдя в состояние ожидания (или же произойдет ошибка, или поток завершится).

В системах с абсолютными приоритетами выполнение активного потока прерывается кроме указанных выше причин, еще при одном условии: если в очереди готовых потоков появился поток, приоритет которого выше приоритета активного потока. В этом случае прерванный поток переходит в состояние готовности (рис. 4.10, б).

Смешанные алгоритмы планирования

Во многих операционных системах алгоритмы планирования построены с использованием как концепции квантования, так и приоритетов. Например, в основе планирования лежит квантование, но величина кванта и/или порядок выбора потока из очереди готовых определяется приоритетами потоков. Именно так реализовано планирование в системе Windows NT, в которой квантование сочетается с динамическими абсолютными приоритетами. На выполнение выбирается готовый поток с наивысшим приоритетом. Ему выделяется квант времени. Если во время выполнения в очереди готовых появляется поток с более высоким приоритетом, то он вытесняет выполняемый поток. Вытесненный поток возвращается в очередь готовых, причем он становится впереди всех остальных потоков имеющих такой же приоритет.

Планирование в системах реального времени

В системах реального времени, в которых главным критерием эффективности является обеспечение временных характеристик вычислительного процесса, планирование имеет особое значение. Любая система реального времени должна реагировать на сигналы управляемого объекта в течение заданных временных ограничений. Необходимость тщательного планирования работ облегчается тем, что в системах реального времени весь набор выполняемых задач известен заранее. Кроме того, часто в системе имеется информация о временах выполнения задач, моментах активизации, предельных допустимых сроках ожидания ответа и т. д. Эти данные могут быть использованы планировщиком для создания статического расписания или для построения адекватного алгоритма динамического планирования.

Моменты перепланировки

Для реализации алгоритма планирования ОС должна получать управление всякий раз, когда в системе происходит событие, требующее перераспределения процессорного времени. К таким событиям могут быть отнесены следующие:

- Прерывание от таймера, сигнализирующее, что время, отведенное активной задаче на выполнение, закончилось. Планировщик переводит задачу в состояние готовности и выполняет перепланирование.
- Активная задача выполнила системный вызов, связанный с запросом на ввод-вывод или на доступ к ресурсу, который в настоящий момент занят (например, файл данных). Планировщик переводит задачу в состояние ожидания и выполняет перепланирование.
- Активная задача выполнила системный вызов, связанный с освобождением ресурса. Планировщик проверяет, не ожидает ли этот ресурс какая-либо задача. Если да, то эта задача переводится из состояния ожидания в состояние готовности. При этом, возможно, что задача, которая получила ресурс, имеет более высокий приоритет, чем текущая активная задача. После перепланирования более приоритетная задача получает доступ к процессору, вытесняя текущую задачу.
- Внешнее (аппаратное) прерывание1, которое сигнализирует о завершении периферийным устройством операции ввода-вывода, переводит соответствующую задачу в очередь готовых, и выполняется планирование.
- Внутреннее прерывание сигнализирует об ошибке, которая произошла в результате выполнения активной задачи. Планировщик снимает задачу и выполняет перепланирование.

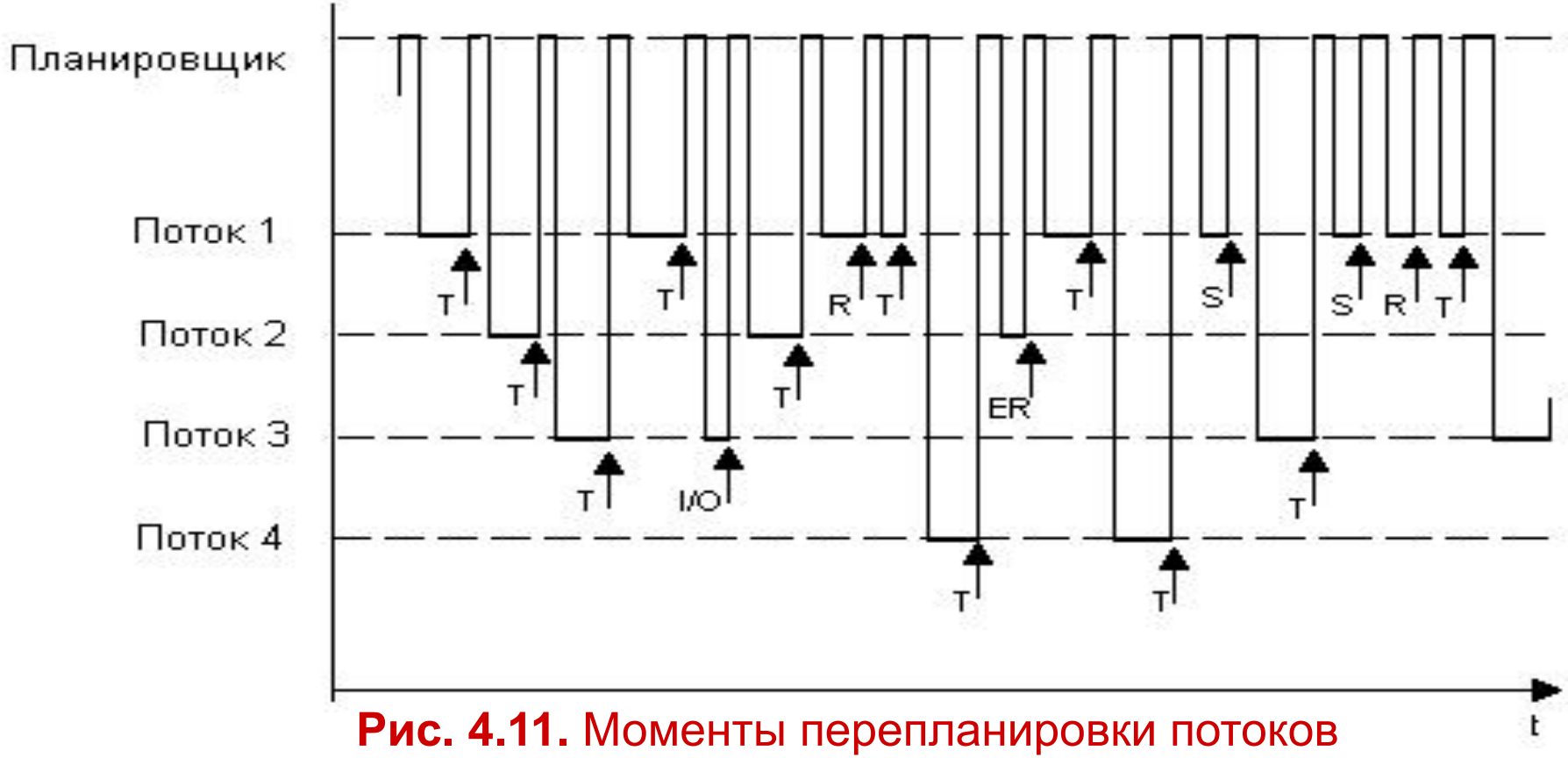

Рис. 4.11. Моменты перепланировки потоков

На рис. 4.11 показан фрагмент временной диаграммы работы планировщика в системе, где одновременно выполняются четыре потока.

Первые четыре цикла работы планировщика, приведенные на рисунке, были инициированы прерываниями от таймера по истечении квантов времени (T).

Следующая передача управления планировщику была осуществлена в результате выполнения потоком 3 системного запроса на ввод-вывод (событие I/O). Планировщик перевел этот поток в состояние ожидания, а затем переключил процессор на поток 2. Поток 2 полностью использовал свой квант, произошло прерывание от таймера, и планировщик активизировал поток 1. При выполнении потока 1 произошло событие R — системный вызов, в результате которого освободился некоторый ресурс (например, был закрыт файл).

Кто Я?

Существуют две альтернативы:

- $Я_{и}$ – *Вечный закон Вечного творения;*
- $я_{л}$ – *временный результат Вечного творения.*

Есть свобода выбора:

Что Человек выбирает тем и становится!

Важнее исследовать базовые принципы организации Операционных Систем чем конкретную её реализация

Пять важных вещей в жизни:

1. Уметь понять и простить.

2. Уважать выбор близкого.

3. Признавать свою вину .

4. Не развивать конфликт, ибо каждая ссора отдаляет людей друг от друга.

5. Не оскорблять родного человека.


(c) 2003 V. Zaikin