Тема 4.1
Многозадачность в операционных системах и методы оптимизации параллельной работы

* ВОПРОСЫ:

1. Многозадачность в ОС. Типы многозадачности.
2. Процессы и потоки.
3. Иерархия, приоритеты и планирование процессов.
4. Менеджер процессов.

1. Многозадачность в ОС. Типы многозадачности.

Многозада́чность (англ. *multitasking*) — свойство операционной системы или среды выполнения обеспечивать возможность параллельной (или псевдопараллельной) обработки нескольких процессов. Истинная многозадачность операционной системы возможна только в распределённых вычислительных системах.

Википедия

Типы многозадачности

Процессная многозадачность (основанная на процессах — одновременно выполняющихся программах). Здесь программа — наименьший элемент кода, которым может управлять планировщик операционной системы. Более известна большинству пользователей (работа в текстовом редакторе и прослушивание музыки).

Поточная многозадачность (основанная на потоках). Наименьший элемент управляемого кода — поток (одна программа может выполнять 2 и более задачи одновременно).

Типы псевдопараллельной многозадачности

Невытесняющая многозадачность
Тип многозадачности, при котором операционная система одновременно загружает в память два или более приложений, но процессорное время предоставляется только основному приложению. Для выполнения фонового приложения оно должно быть активизировано.

Подобная многозадачность может быть реализована не только в операционной системе, но и с помощью программ-переключателей задач.

Типы псевдопараллельной многозадачности

Совместная или кооперативная многозадачность
Тип многозадачности, при котором следующая задача выполняется только после того, как текущая задача явно объявит себя готовой отдать процессорное время другим задачам. (Windows версий до 3.x включительно, а также 16-битные приложения в Windows 9x).

Кооперативную многозадачность можно назвать многозадачностью «второй ступени».

При простом переключении активная программа получает все процессорное время, а фоновые приложения полностью замораживаются. При кооперативной многозадачности приложение может захватить фактически столько процессорного времени, сколько оно считает нужным. Все приложения делят процессорное время, периодически передавая управление следующей задаче.

Типы псевдопараллельной многозадачности

Вытесняющая, или приоритетная, многозадачность
Вид многозадачности, в котором операционная система сама передает управление от одной выполняемой программы другой в случае завершения операций ввода-вывода, возникновения событий в аппаратуре компьютера, истечения таймеров и квантов времени, или же поступлений тех или иных сигналов от одной программы к другой. В этом виде многозадачности процессор может быть переключен с исполнения одной программы на исполнение другой без всякого пожелания первой программы и буквально между любыми двумя инструкциями в её коде. Распределение процессорного времени осуществляется планировщиком процессов.

Типы псевдопараллельной многозадачности

Вытесняющая, или приоритетная, многозадачность

Преимущества:

- Возможность полной реализации многозадачного ввода-вывода в ядре ОС, когда ожидание завершения ввода-вывода одной программой позволяет процессору тем временем исполнять другую программу.
- Сильное повышение надежности системы в целом, в сочетании с использованием защиты памяти — идеал в виде «ни одна программа пользовательского режима не может нарушить работу ОС в целом» становится достижимым хотя бы теоретически, вне вытесняющей многозадачности он не достижим даже в теории.
- Возможность полного использования многопроцессорных и многоядерных систем.

Типы псевдопараллельной многозадачности

Вытесняющая, или приоритетная, многозадачность

Недостатки:

- необходимость особой дисциплины при написании кода,
- особые требования к защите всех разделяемых и глобальных данных объектами

2. Процессы и потоки.

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».**

Сложности

1. Специфика различных ОС, когда совпадающие по сути понятия получили разные названия, например задача (task) в OS/2, OS/360 и процесс (process) в UNIX, Windows NT, NetWare.
2. Некоторые из терминов получили новое смысловое значение, особенно это касается понятия «процесс», который уступил многие свои свойства новому понятию «поток».
3. Терминологические сложности порождаются наличием нескольких вариантов перевода англоязычных терминов на русский язык. Например, термин «thread» переводится как «нить», «поток», «облегченный процесс», «минизадача» и др.

2. Процессы и потоки.

Подсистема управления процессами и потоками ОС занимается

- их созданием и уничтожением,
- поддерживает взаимодействие между ними,
- распределяет процессорное время между несколькими одновременно существующими в системе процессами и потоками.

2. Процессы и потоки.

В чем же состоят принципиальные отличия в понятиях «процесс» и «поток»?

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 и получили свое название благодаря тому, что они представляют собой последовательности (потоки выполнения) команд.

2. Процессы и потоки.

Для того чтобы процессы не могли вмешаться в распределение ресурсов, а также не могли повредить коды и данные друг друга, важнейшей задачей ОС является изоляция одного процесса от другого. Для этого операционная система обеспечивает каждый процесс отдельным виртуальным адресным пространством, так что ни один процесс не может получить прямого доступа к командам и данным другого процесса.

ПРИМЕЧАНИЕ

Виртуальное адресное пространство процесса — это совокупность адресов, которыми может манипулировать программный модуль процесса. Операционная система отображает виртуальное адресное пространство процесса на отведенную процессу физическую память.

2. Процессы и потоки.

Потоки возникли в операционных системах как средство распараллеливания вычислений.

Задача распараллеливания вычислений в рамках одного приложения может быть решена и традиционными способами:

- прикладной программист может взять на себя сложную задачу организации параллелизма, выделив в приложении некоторую подпрограмму- диспетчер, которая периодически передает управление той или иной ветви вычислений.
- создание для одного приложения нескольких процессов для каждой из параллельных работ.

Но это неэффективно!

2. Процессы и потоки.

В операционной системе наряду с процессами нужен другой механизм распараллеливания вычислений, который учитывал бы тесные связи между отдельными ветвями вычислений одного и того же приложения. Для этих целей современные ОС предлагают механизм многопоточной обработки (multithreading). При этом вводится новая единица работы — поток выполнения, а понятие «процесс» в значительной степени меняет смысл.

Понятию «поток» соответствует последовательный переход процессора от одной команды программы к другой. ОС распределяет процессорное время между потоками. Процессу ОС назначает адресное пространство и набор ресурсов, которые совместно используются всеми его потоками.

2. Процессы и потоки.

В отличие от процессов, которые принадлежат разным конкурирующим приложениям, 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. Это означает, что они разделяют одни и те же глобальные переменные.

Потоки разных процессов хорошо защищены друг от друга.

2. Процессы и потоки.

Мультипрограммирование более эффективно на уровне потоков, а не процессов.

Каждый поток имеет собственный счетчик команд и стек. Задача, оформленная в виде нескольких потоков в рамках одного процесса, может быть выполнена быстрее за счет псевдопараллельного (или параллельного в мультипроцессорной системе) выполнения ее отдельных частей.

Эффекты

Например,

- в задачах типа «писатель-читатель» один поток выполняет запись в буфер, а другой считывает записи из него. Поскольку они разделяют общий буфер, не стоит их делать отдельными процессами.

- управление сигналами, такими как прерывание с клавиатуры (del или break). Вместо обработки сигнала прерывания один поток назначается для постоянного ожидания поступления сигналов. Таким образом, использование потоков может сократить необходимость в прерываниях пользовательского уровня.

Создание процессов и потоков

Создать процесс — это прежде всего означает создать описатель процесса, в качестве которого выступает одна или несколько информационных структур, содержащих все сведения о процессе, необходимые операционной системе для управления им. В число таких сведений могут входить, например, идентификатор процесса, данные о расположении в памяти исполняемого модуля, степень привилегированности процесса (приоритет и права доступа) и т. п. Примерами описателей процесса являются 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.

Создание процессов и потоков

Создание процесса включает **загрузку кодов и данных исполняемой программы данного процесса с диска в оперативную память.** Для этого ОС должна обнаружить местоположение такой программы на диске, перераспределить оперативную память и выделить память исполняемой программе нового процесса.

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.

Создание процессов и потоков

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

Создание процессов и потоков

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.

Создание процессов и потоков

В исходном состоянии поток находится в приостановленном состоянии.
Момент выборки потока на выполнение осуществляется в соответствии с принятым в данной системе правилом предоставления процессорного времени и с учетом всех существующих в данный момент потоков и процессов.

В случае если коды и данные процесса находятся в области подкачки, необходимым условием активизации потока процесса является также наличие места в оперативной памяти для загрузки его исполняемого модуля.

Создание процессов и потоков

Поток может обратиться к ОС с запросом на создание так называемых **потоков-потомков**. В разных ОС по-разному строятся отношения между потоками-потомками и их родителями. Например, в одних ОС выполнение родительского потока синхронизируется с его потомками, в частности после завершения родительского потока ОС может снимать с выполнения всех его потомков. В других системах потоки-потомки могут выполняться асинхронно по отношению к родительскому потоку. Потомки, как правило, наследуют многие свойства родительских потоков. Во многих системах порождение потомков является основным механизмом создания процессов и потоков.

3. Иерархия, приоритеты и планирование процессов.

При управлении процессами операционная система использует два основных типа информационных структур: дескриптор процесса и контекст процесса. Дескриптор процесса содержит такую информацию о процессе, которая необходима ядру в течение всего жизненного цикла процесса независимо от того, находится он в активном или пассивном состоянии, находится образ процесса в оперативной памяти или выгружен на диск.

(Образом процесса называется совокупность его кодов и данных.)

3. Иерархия, приоритеты и планирование процессов.

Дескрипторы отдельных процессов объединены в список, образующий таблицу процессов. Память для таблицы процессов отводится динамически в области ядра. На основании информации, содержащейся в таблице процессов, операционная система осуществляет планирование и синхронизацию процессов.

В дескрипторе прямо или косвенно (через указатели, на связанные с процессом структуры) содержится информация о состоянии процесса, о расположении образа процесса в оперативной памяти и на диске, о значении отдельных составляющих приоритета, а также о его итоговом значении — глобальном приоритете, об идентификаторе пользователя, создавшего процесс, о родственных процессах, о событиях, осуществления которых ожидает данный процесс, и некоторая другая информация.

3. Иерархия, приоритеты и планирование процессов.

Контекст процесса содержит менее оперативную, но более объемную часть информации о процессе, необходимую для возобновления выполнения процесса с прерванного места:

- содержимое регистров процессора,
- коды ошибок выполняемых процессором системных вызовов,
- информация обо всех открытых данным процессом файлах и незавершенных операциях ввода-вывода,
- другие данные, характеризующие состояние вычислительной среды в момент прерывания.

3. Иерархия, приоритеты и планирование процессов.

в UNIX порождение нового процесса происходит в два этапа —

- сначала создается копия процесса-родителя,
- затем у нового процесса производится замена кодового сегмента на заданный.

Вновь созданному процессу операционная система присваивает целочисленный идентификатор, уникальный на весь период функционирования системы.

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.

Планирование и диспетчеризация потоков

Планирование потоков включает в себя решение двух задач:

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

Планирование и диспетчеризация потоков

Диспетчеризация заключается в реализации найденного в результате планирования (динамического или статистического) решения, то есть в переключении процессора с одного потока на другой.

Диспетчеризация сводится к следующему:

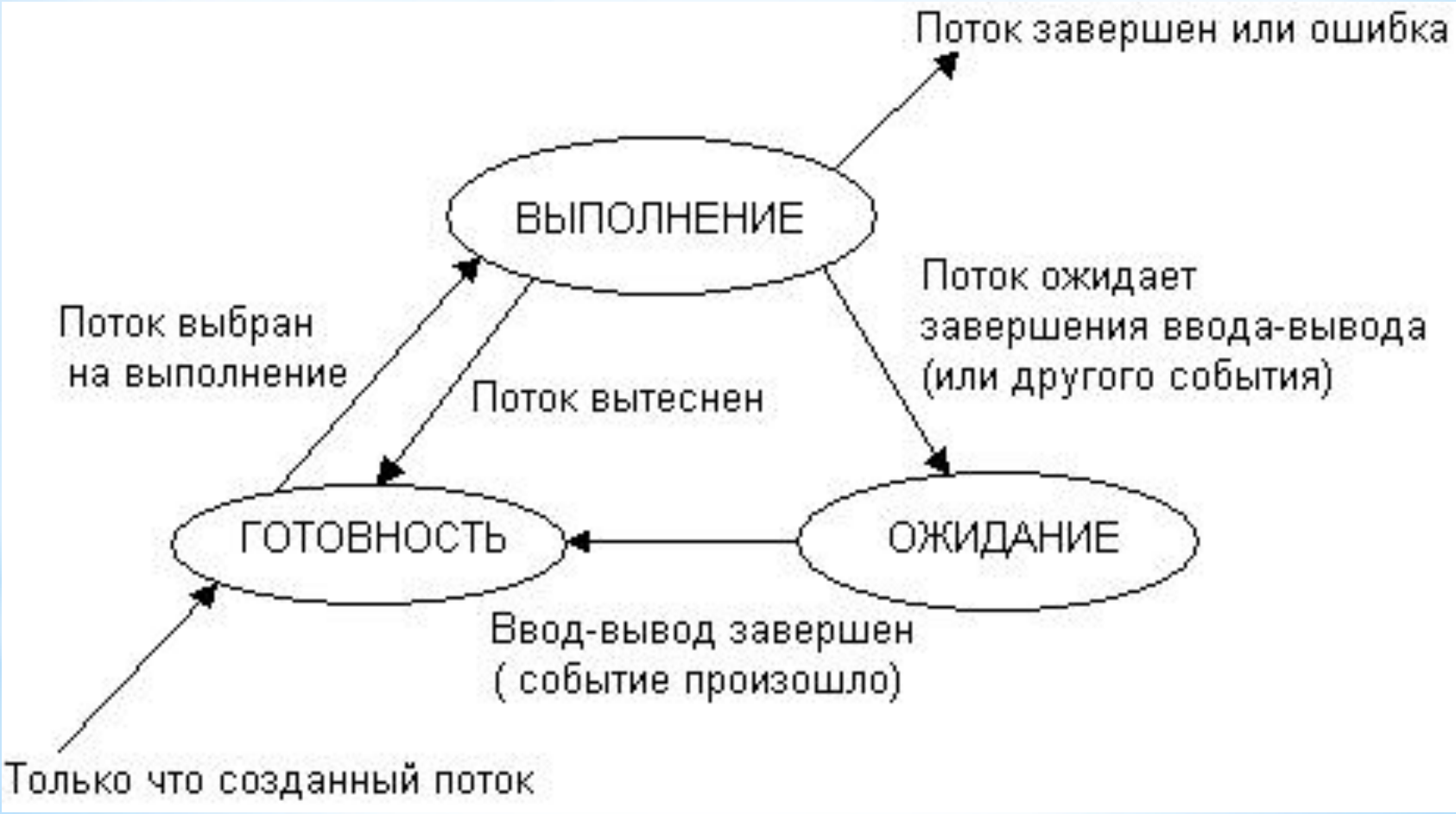
- сохранение контекста текущего потока, который требуется сменить;
- загрузка контекста нового потока, выбранного в результате планирования;
- запуск нового потока на выполнение.

3. Иерархия, приоритеты и планирование процессов.

В мультипрограммной системе поток может находиться в одном из трех основных состояний:

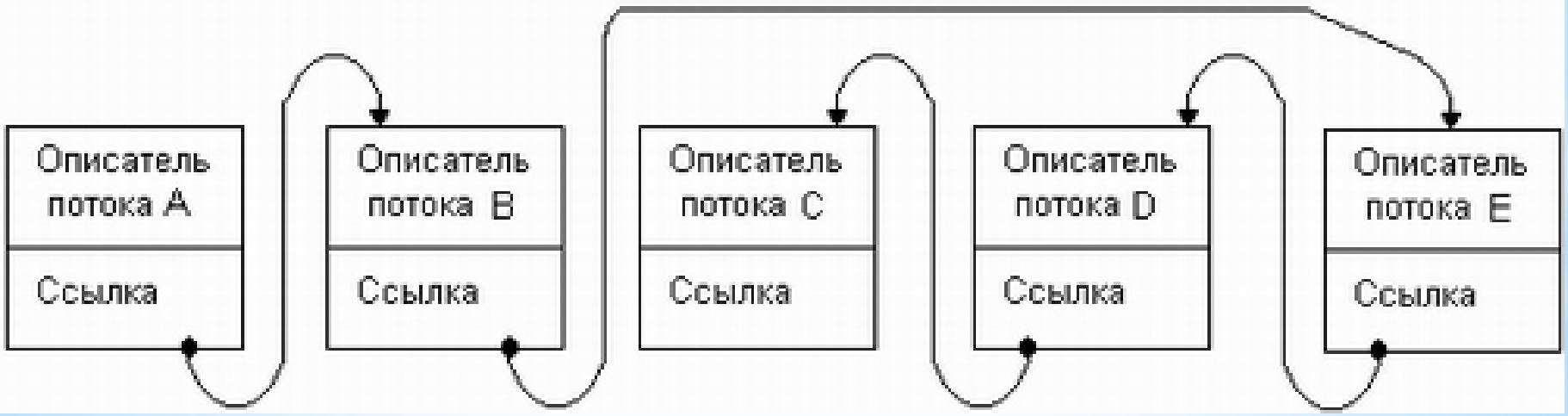
1. выполнение — активное состояние потока, во время которого поток обладает всеми необходимыми ресурсами и непосредственно выполняется процессором;
2. ожидание — пассивное состояние потока, находясь в котором, поток заблокирован по своим внутренним причинам (ждет осуществления некоторого события, например завершения операции ввода-вывода, получения сообщения от другого потока или освобождения какого-либо необходимого ему ресурса);
3. 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).

В мультипрограммной системе поток может находиться в одном из трех основных состояний:



3. Иерархия, приоритеты и планирование процессов.

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.



Запланированный порядок выполнения выглядит так: A, B, E, D, C

3. Иерархия, приоритеты и планирование процессов.

Все множество алгоритмов планирования можно разделить на два класса: вытесняющие и невытесняющие алгоритмы планирования.

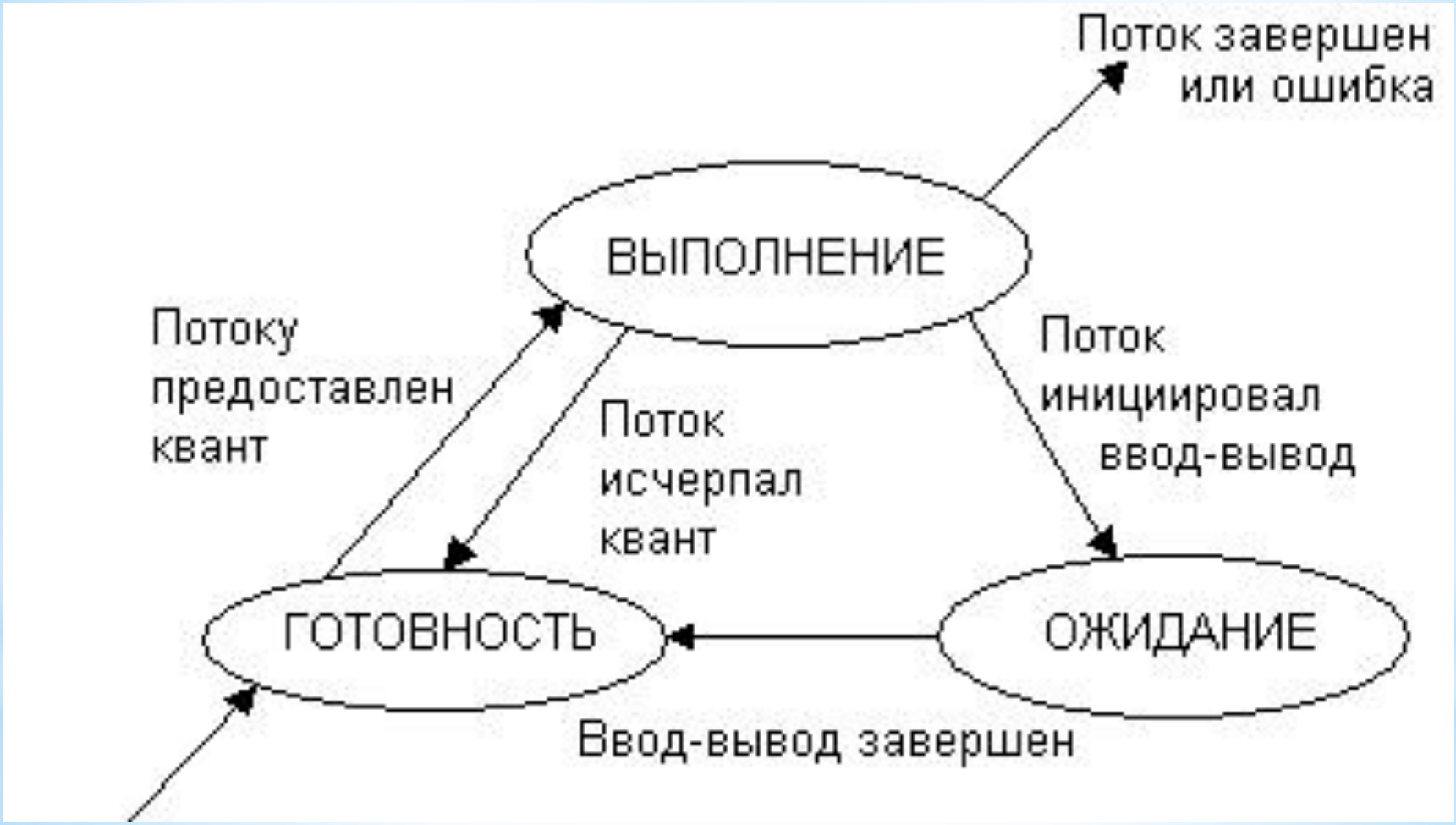
- Невытесняющие (non-preemptive) алгоритмы основаны на том, что активному потоку позволяется выполняться, пока он сам, по собственной инициативе, не отдаст управление операционной системе для того, чтобы та выбрала из очереди другой готовый к выполнению поток.
- Вытесняющие (preemptive) алгоритмы — это 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.

3. Иерархия, приоритеты и планирование процессов.

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:

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

Концепция квантования



Концепция квантования

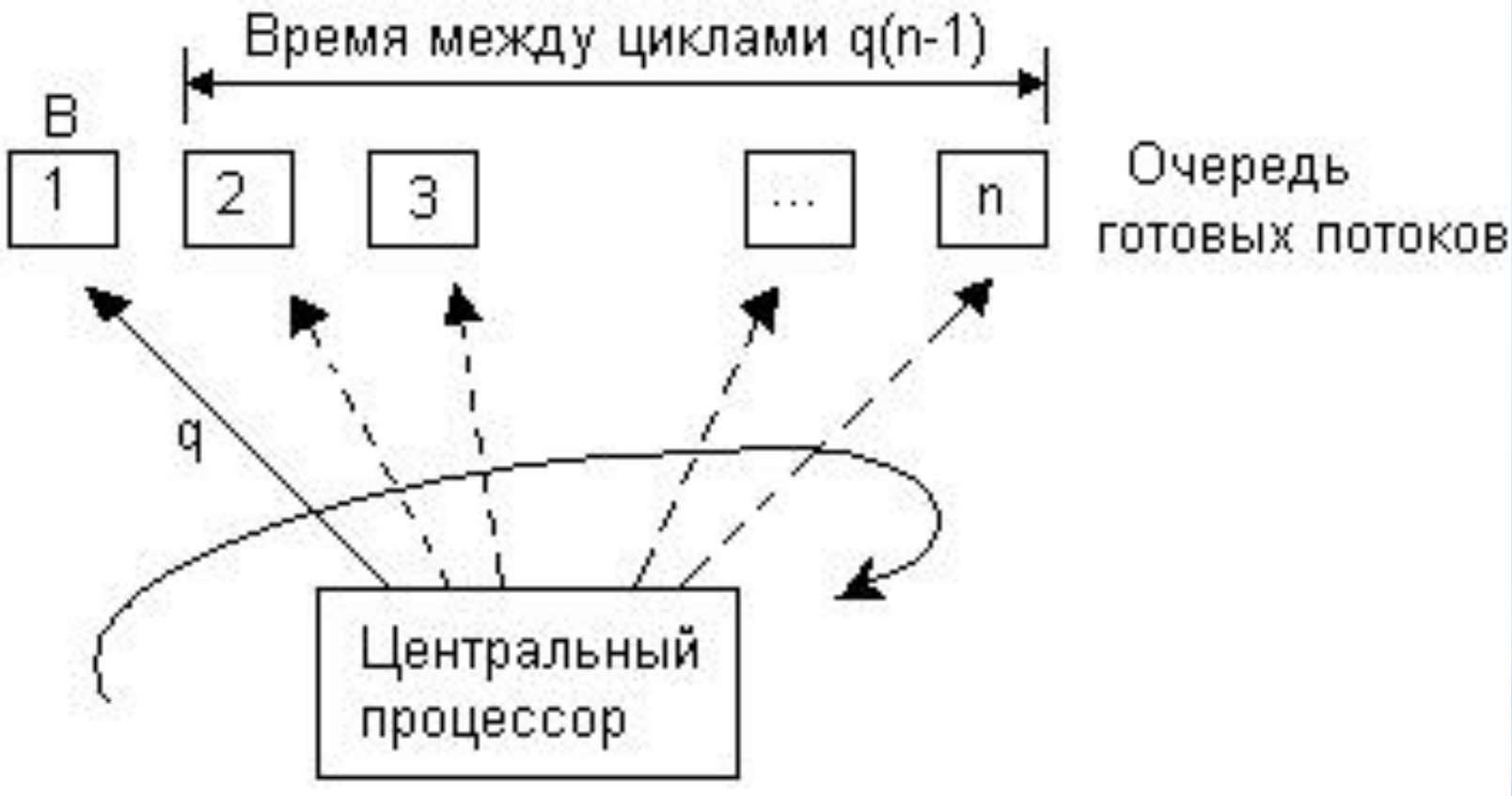
Кванты, выделяемые потокам, могут быть одинаковыми для всех потоков или различными.

1 случай: всем потокам предоставляются кванты одинаковой длины q.

Если в системе имеется N потоков, то время, которое поток проводит в ожидании следующего кванта, можно грубо оценить как q(N-1).

Если величина кванта выбрана очень небольшой, то значение произведения q(n-1) 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.

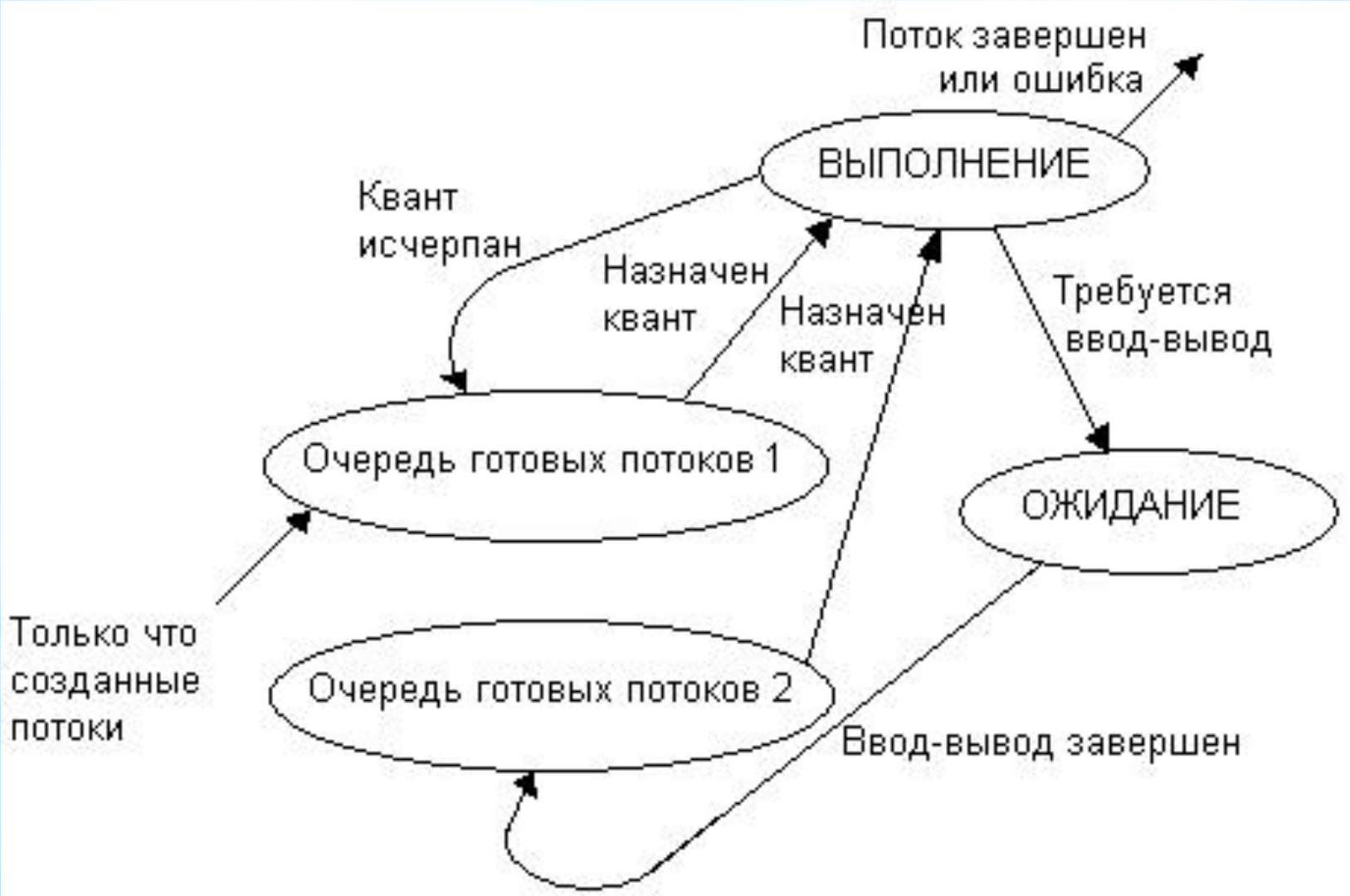
Концепция квантования



Концепция квантования

Потоки получают для выполнения квант времени, но некоторые из них используют его не полностью, например, из-за необходимости выполнить ввод или вывод данных. В результате возникает ситуация, когда потоки с интенсивными обращениями к вводу-выводу используют только небольшую часть выделенного им процессорного времени. Алгоритм планирования может исправить эту «несправедливость». В качестве компенсации за неиспользованные полностью кванты потоки получают привилегии при последующем обслуживании. Для этого планировщик создает две очереди готовых потоков.

Концепция квантования



Концепция приоритетного обслуживания

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.

Концепция приоритетного обслуживания

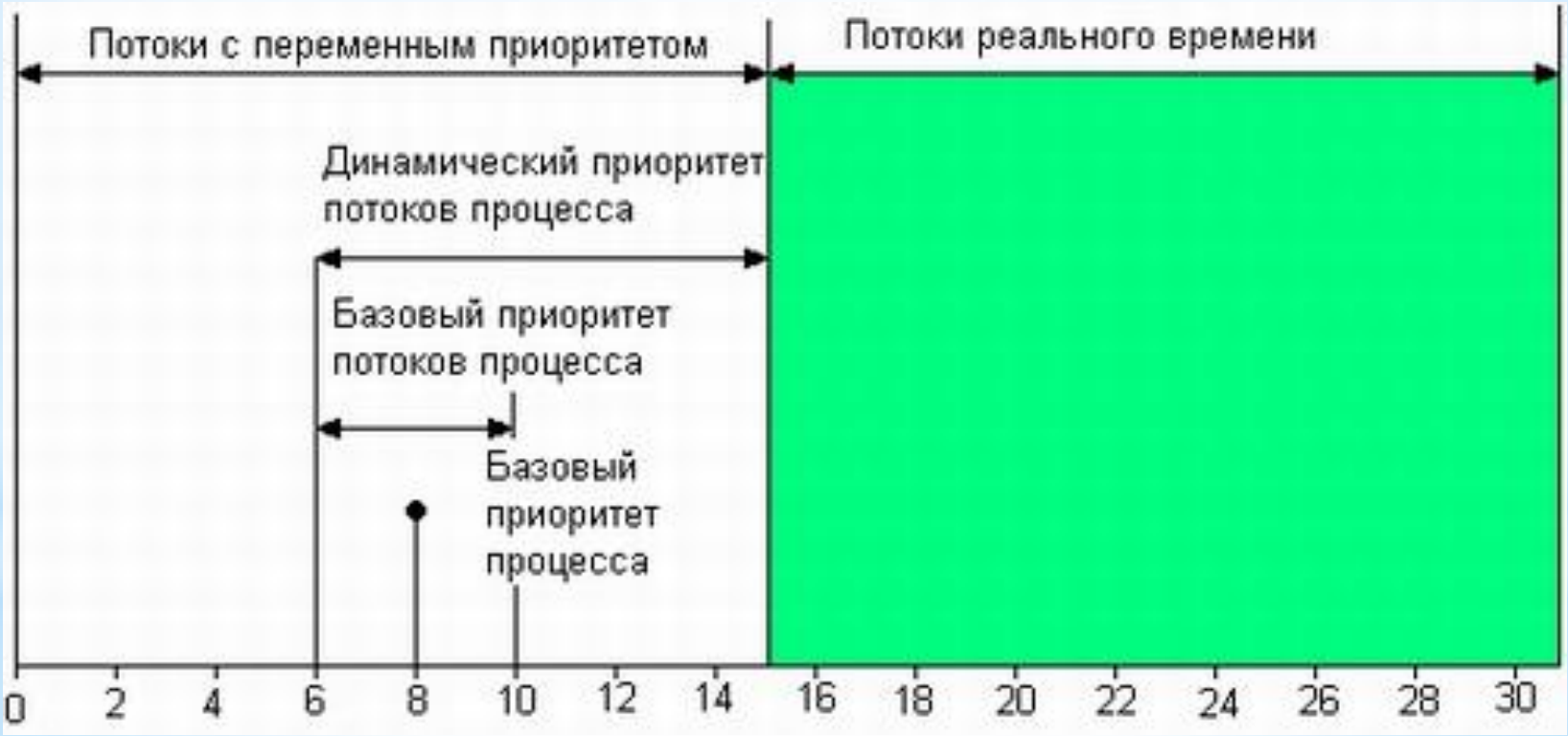
Приоритет может выражаться целым или дробным, положительным или отрицательным значением.

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.

Концепция приоритетного обслуживания

Схема назначения приоритетов потокам, принятую в операционной системе Windows NT



Концепция приоритетного обслуживания

Существуют две разновидности приоритетного планирования:

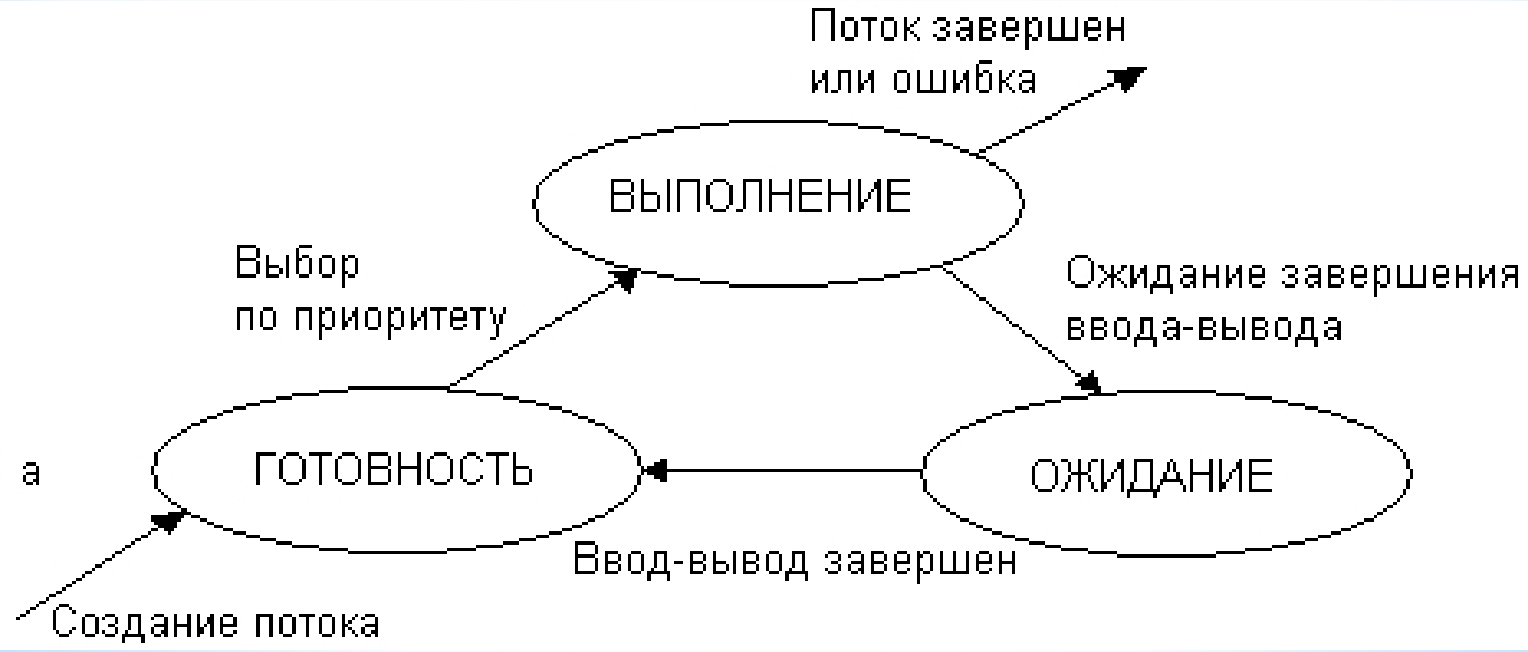
- обслуживание с относительными приоритетами,
- обслуживание с абсолютными приоритетами.

В обоих случаях выбор потока на выполнение из очереди готовых осуществляется одинаково: выбирается поток, имеющий наивысший приоритет.

Момент смены активного потока решается по-разному.

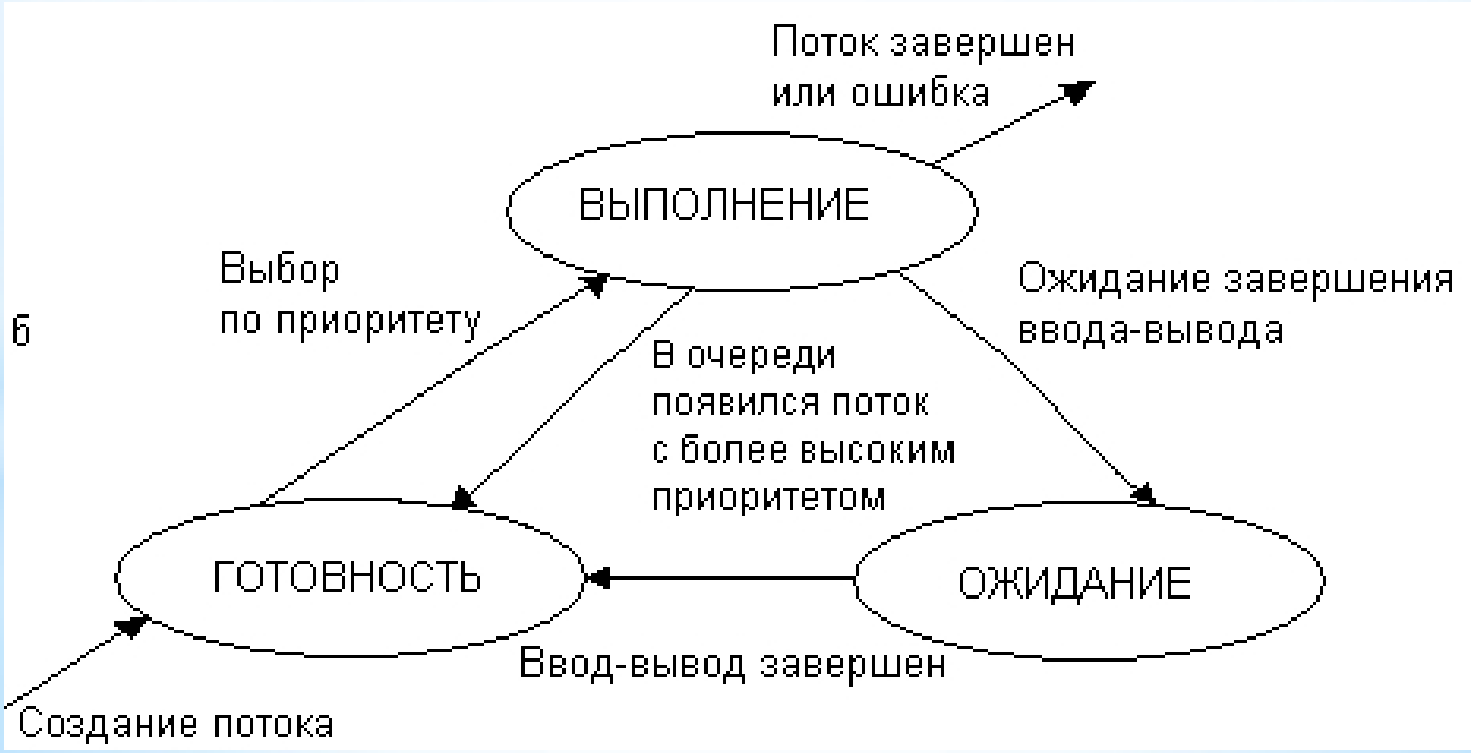
Концепция приоритетного обслуживания

В системах с относительными приоритетами 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 пакетной обработки (в том числе известной ОС OS/360) относительные приоритеты используются широко.



Концепция приоритетного обслуживания

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



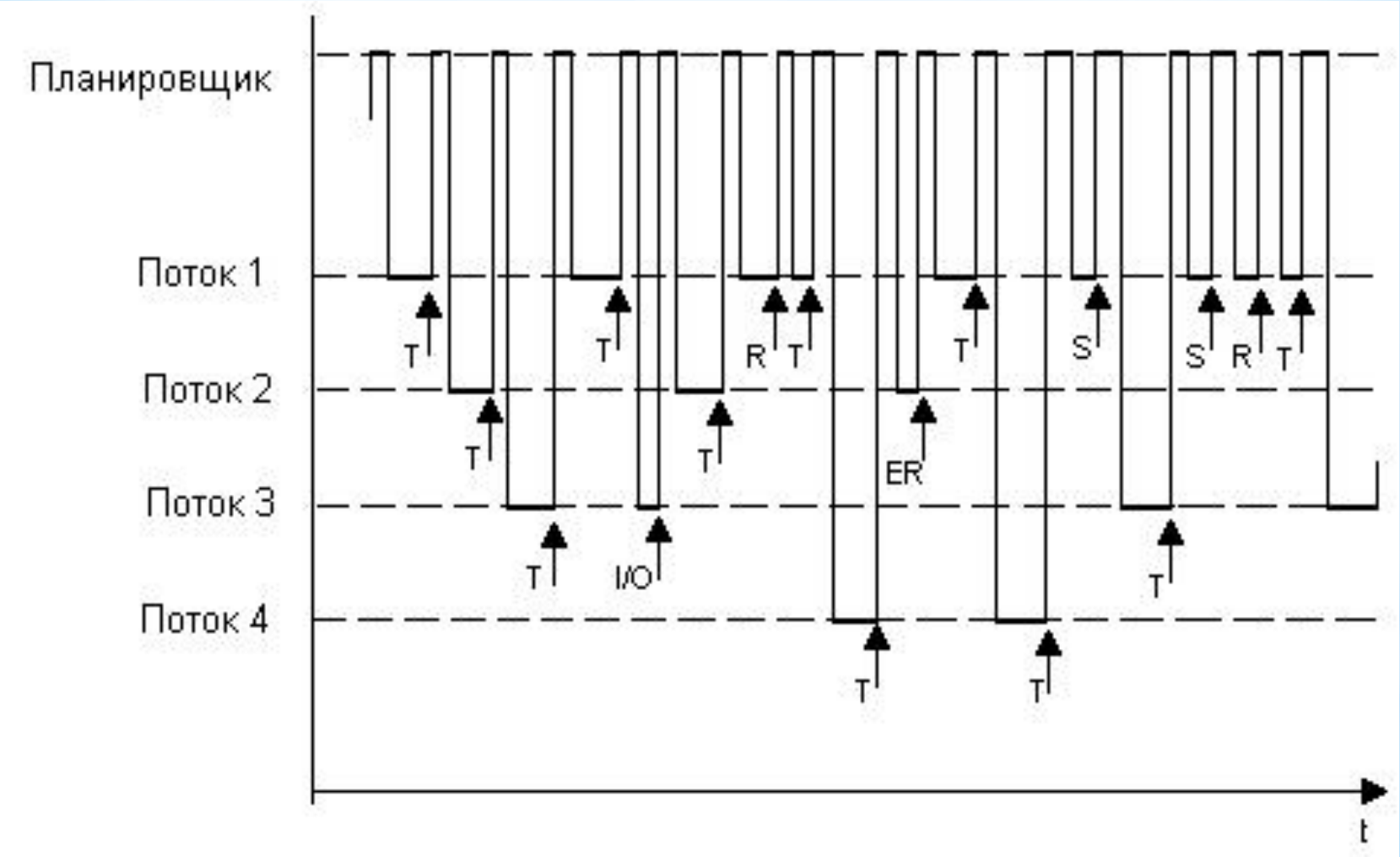
Концепция приоритетного обслуживания

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.
В операционной системе UNIX System V Release 4 понятие «поток» отсутствует, и планирование осуществляется на уровне процессов. В системе UNIX System V Release 4 реализована вытесняющая многозадачность, основанная на использовании приоритетов и квантования.

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



Моменты перепланировки

Первые четыре цикла работы планировщика, приведенные на рисунке, были инициированы прерываниями от таймера по истечении квантов времени (эти события обозначены на рисунке как 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). Это событие вызвало перепланировку потоков. Планировщик просмотрел очередь ожидающих потоков и обнаружил, что поток 4 ждет освобождения данного ресурса. Этот поток был переведен в состояние готовности, но поскольку приоритет выполняющегося в данный момент потока 1 выше приоритета потока 4, планировщик вернул процессор потоку 1.

Моменты перепланировки

В следующем цикле работы планировщик активизировал поток 4, а затем, после истечения кванта и сигнала от таймера, управление получил поток 2. Этот поток не успел использовать свой квант, так как был снят с выполнения в результате возникшей ошибки (событие ER).
Далее планировщик предоставлял процессорное время потокам 1, 4 и снова 1. Во время выполнения потока 1 произошло прерывание S от внешнего устройства, сигнализирующее о том, что операция передачи данных завершена. Это событие активизировало работу планировщика, в результате которой поток 3, ожидавший завершения ввода-вывода, вытеснил поток 1, так как имел в этот момент более высокий приоритет. Последний показанный на диаграмме период выполнения потока 1 прерывался несколько раз. Вначале это было прерывание от внешнего устройства (S), затем программное прерывание (R), вызвавшее освобождение ресурса, и, наконец, прерывание от таймера (T). Каждое из этих трех прерываний вызвало перепланировку потоков. В двух первых случаях планировщик оставил выполняться поток 1, так как в очереди не оказалось более приоритетных потоков, а квант времени, выделенный потоку 1, еще не был исчерпан. Переключение потоков было выполнено только по прерыванию от таймера.

4. Менеджер процессов.

Запросы многих пользователей по коррекции процессов вполне удовлетворяет стандартный «Диспетчер задач». (OS Windows)

Штатный «Диспетчер задач» большого массива сведений об активных в данный момент программах не предоставляет и не дает оценки их безвредности.

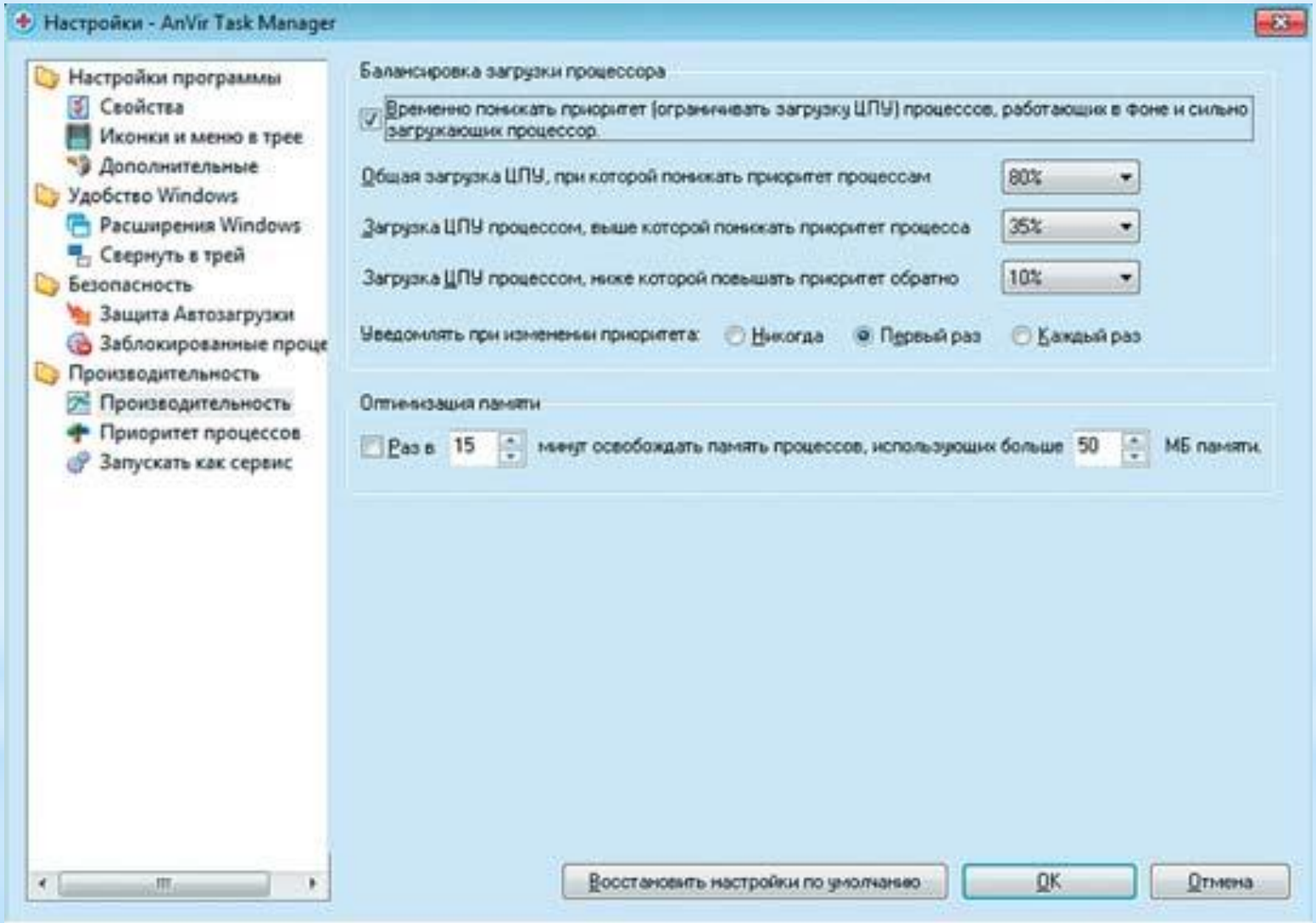
4. Менеджер процессов.

AnVir Task Manager 6.5.0
Разработчик: AnVir Software
Веб-сайт: www.anvir.net
Размер дистрибутива: 3,97 Мбайт
Условия распространения: Free for noncommercial use (для систем с русским языком)

Помимо собственно менеджера процессов, входит утилита быстрой отправки сомнительных, на ваш взгляд, файлов для проверки онлайновым антивирусным сервисом на сайте **www.virustotal.com**, монитор центрального процессора Core Temp, а также небольшой оптимизатор ОС Windows — Tweaker, с помощью которого вы сможете настроить основные параметры системы, а также скрыть при необходимости отдельные ее компоненты.

4. Менеджер процессов.

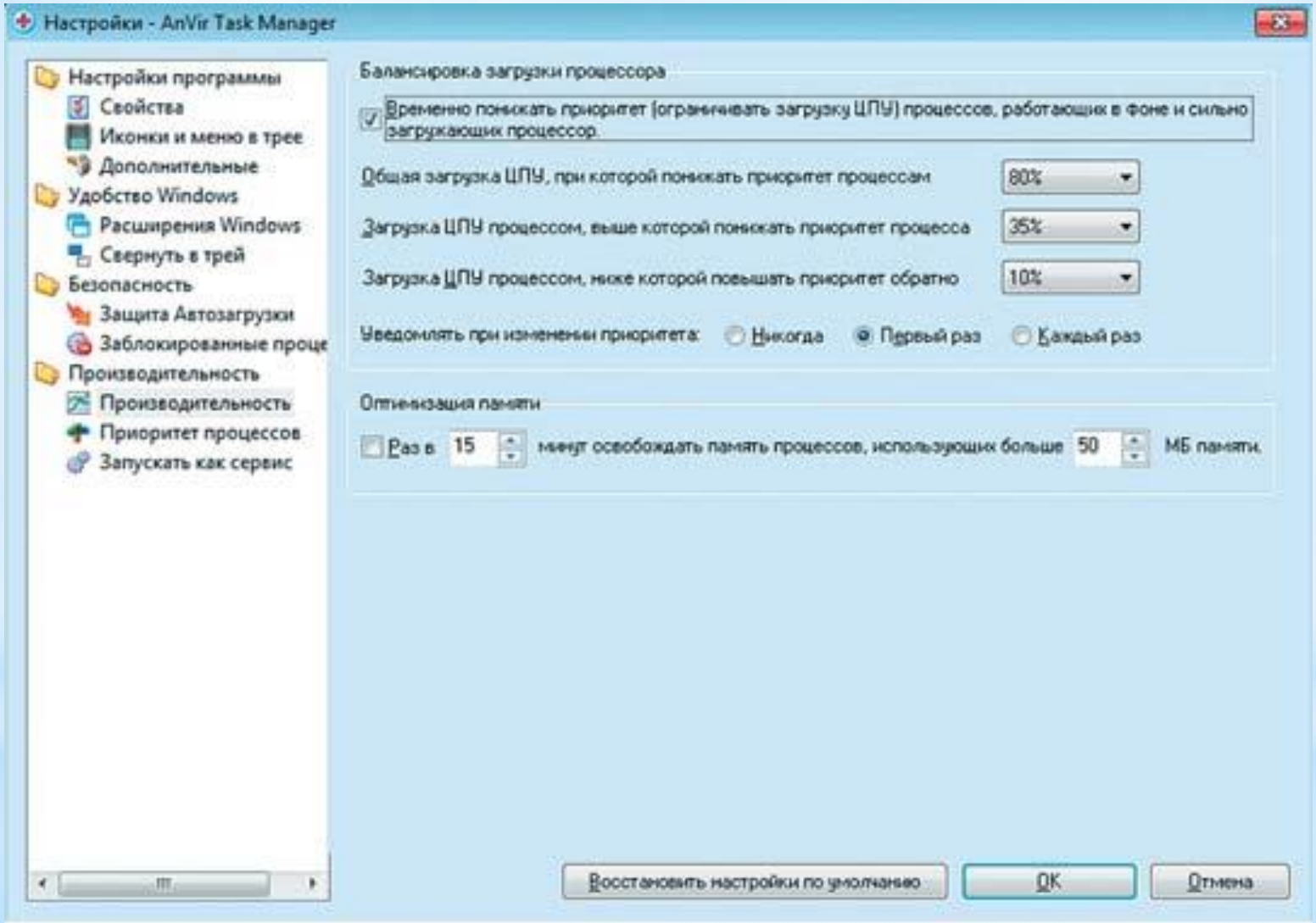
AnVir Task Manager 6.5.0



В «Настройках» менеджера процессов AnVir Task Manager можно задать автоматическое понижение приоритета для фоновых процессов

4. Менеджер процессов.

AnVir Task Manager 6.5.0



В «Настройках» менеджера процессов AnVir Task Manager можно задать автоматическое понижение приоритета для фоновых процессов

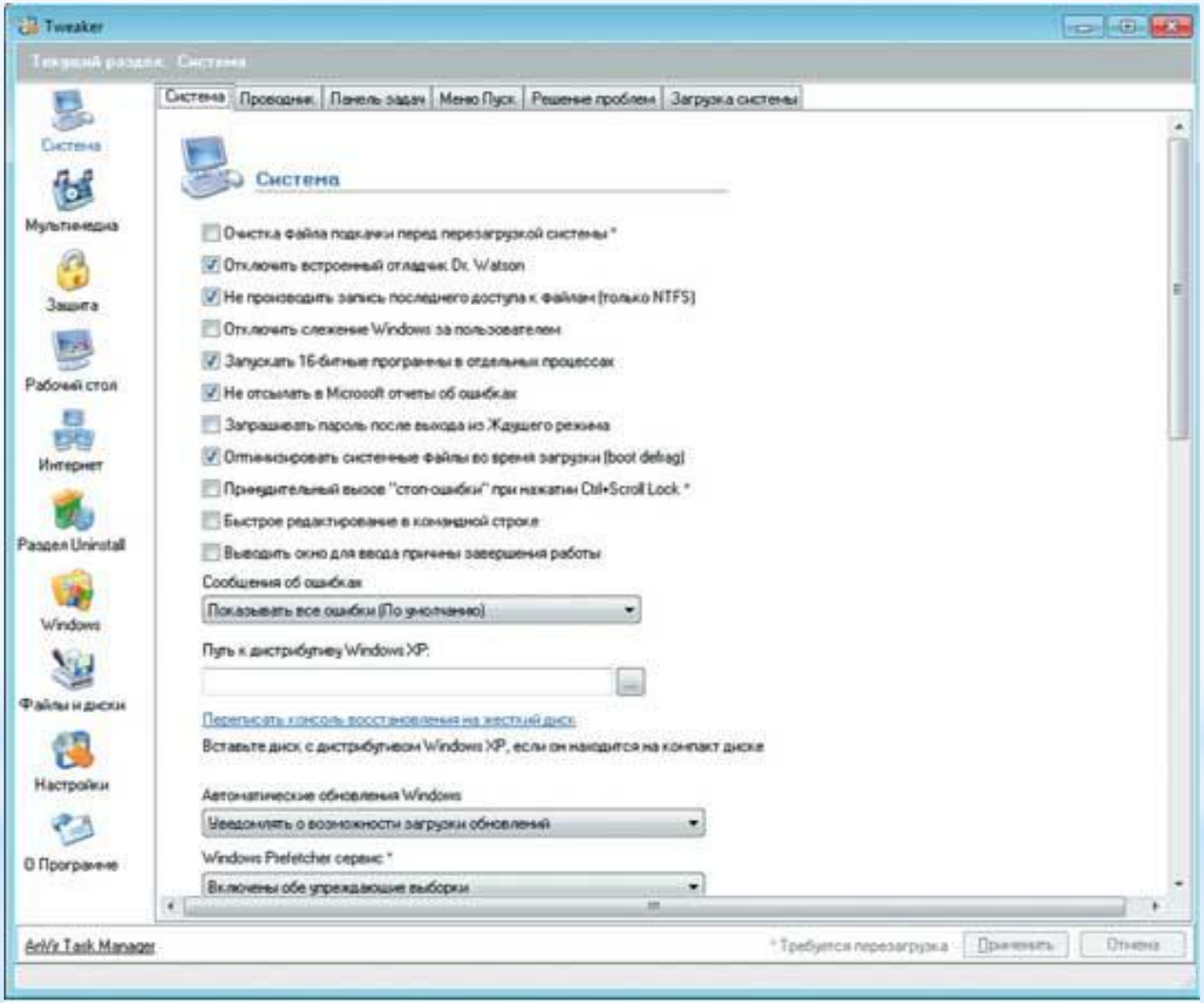
AnVir Task Manager 6.5.0

Есть возможность быстро убедиться в надежности или, наоборот, в наличии потенциальной угрозы в любом процессе, список которых программа AnVir Task Manager демонстрирует на странице Processes.

- Вы сможете отключить любую активную в настоящее время программу или сервис, нажав кнопку «Завершить процесс».
- С помощью пункта «Приостановить процесс» в контекстном меню можно временно заморозить ту или иную утилиту.
- Имеется также команда постоянной блокировки — «Добавить к заблокированным (Карантин)».

4. Менеджер процессов.

AnVir Task Manager 6.5.0



Менеджер процессов AnVir Task Manager предоставляет в наше распоряжение еще и встроенный твикер ОС Windows

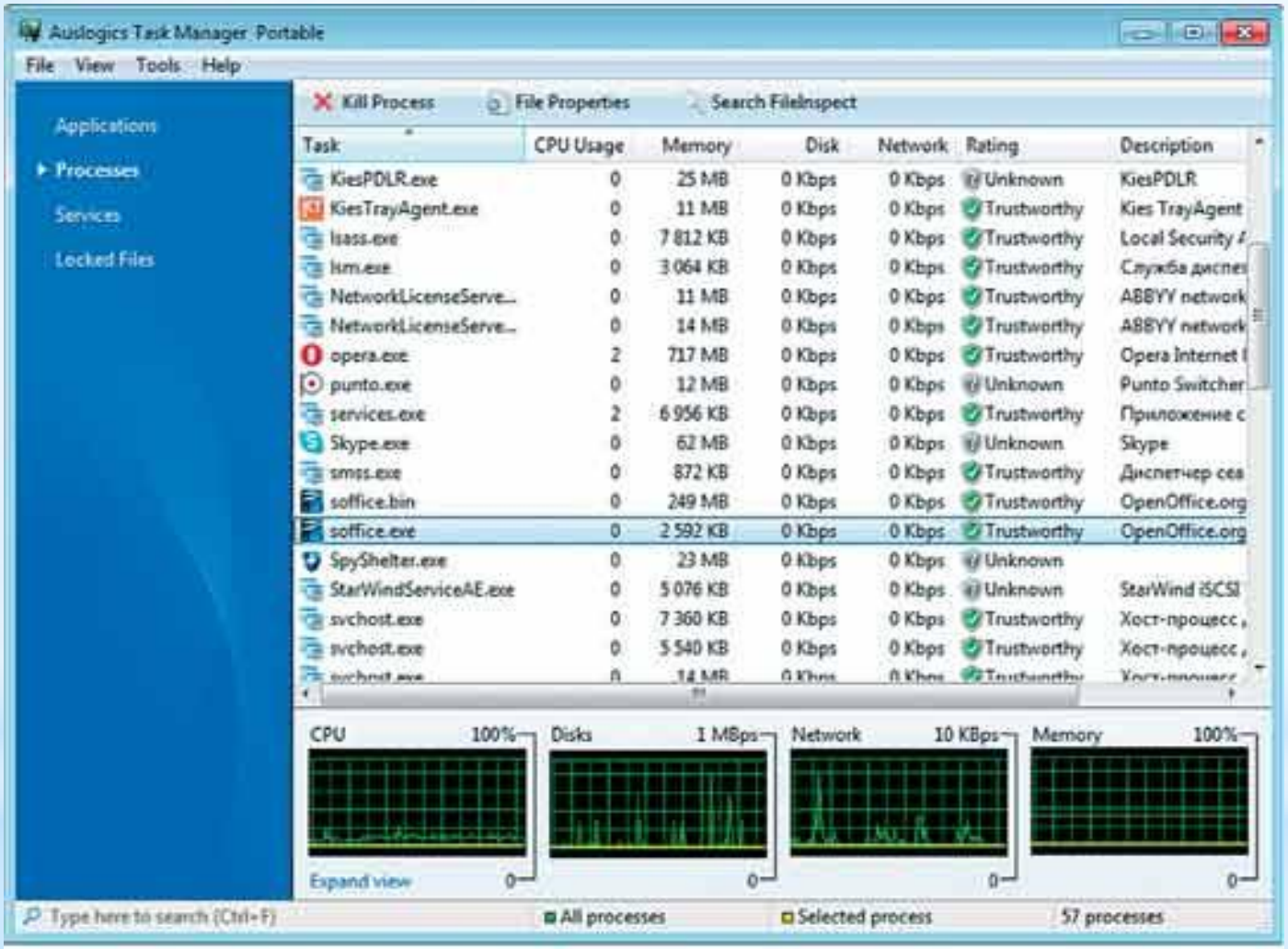
4. Менеджер процессов.

Auslogics Task Manager Portable 2.2.0.0
Разработчик: Auslogics
Веб-сайт: www.anvir.net
Размер дистрибутива: 3,97 Мбайт
Условия распространения: Free for personal use

В составе программы **Auslogics Task Manager**, представленной, как и AnVir Task Manager, в том числе Portable-версией, нет никаких вспомогательных утилит — она занимается только управлением активными процессами.

4. Менеджер процессов.

Auslogics Task Manager Portable 2.2.0.0



Программа Auslogics Task Manager Portable сообщает пользователю, можно ли доверять тому или иному приложению

Auslogics Task Manager Portable 2.2.0.0

Об известных же файлах программа поможет получить подробную справку на сайте www.fileinspect.com.

Чтобы открыть в браузере по умолчанию страницу на этом веб-сервисе с интересующим приложением, выберите кнопку Search FileInspect.

4. Менеджер процессов.

Process Hacker 2.27
Разработчик: wj32
Веб-сайт: processhacker.sourceforge.net
Размер дистрибутива: 2,15 Мбайт
Условия распространения: Open Source
О возможной потере информации при попытке закрыть тот или иной процесс известит вас и программа **Process Hacker**. Эта утилита обладает большим арсеналом средств для отключения зависших и просто бесполезных приложений в разделе Processes.

Process Hacker 2.27

Можно при содействии Process Hacker выявить скрытые процессы.

Те из них, что обозначены красным цветом, являются скрытыми. Если таковые будут обнаружены, необходимо проявить внимание на предмет наличия вредоносного кода. Имеется возможность прямо в данном окне их выгрузить, нажав кнопку Terminate.

На закладке Services Process Hacker продемонстрирует имеющиеся в системе службы. Любые из них можно на свое усмотрение приостановить или запустить.

Если нужно узнать, как работают во Всемирной паутине те или программы, — раздел Network. При желании с помощью команды Go to Process контекстного меню можно перейти обратно на страницу Processes, причем выбрано в ней будет как раз то приложение, которое заинтересовало вас своей сетевой деятельностью.

4. Менеджер процессов.

Некоммерческая утилита ESET SysInspector (www.esetnod32.ru/download/other/sysin-spector) известного производителя анти- вирусного ПО компании ESET.

Данное приложение при запуске собирает сведения обо всем, что в настоящий момент происходит в ОС, в том числе об активных процессах, системных службах, драйверах, и предоставляет эту информацию пользователю. Вы можете сохранить ее в виде лога, предоставляющего собой ZIP-архив (File / Save Log). Потом, когда будет нужно, вы сможете сопоставить два состояния системы — текущее и то, что было ранее сохранено (File / Compare Logs / Select File). При этом можно отдельно увидеть, что было добавлено (Added), удалено (Removed), заменено (Replaced).

4. Менеджер процессов.

Некоммерческая утилита ESET SysInspector

Мониторинг и убийство процессов в Ubuntu

Способ первый, графический

Щелкаем правой кнопкой на нижней панели и выбираем пункт "Добавить на панель...", ищем апплет "Системный монитор" и добавляем его кнопкой "Добавить" или перетаскиванием на панель.
Щелкнув правой кнопкой по "Системный монитор" и выбрав пункт "Параметры" можно настроить несколько графиков для отображения активности процессора, памяти, сети, жестких дисков и так далее, таким образом вы всегда будете видеть как на ладони - чем занимается ваш компьютер.

Мониторинг и убийство процессов в Ubuntu

Способ второй - в консоли.

Чтобы перейти в консоль, нажмите CTRL+ALT+F1 - вы попадете в первую консоль, CTRL+ALT+F2 - во вторую и т.д. Графические консоли обычно начинаются с 7 или 8 - так вы сможете вернуться в графику.
Первое, что необходимо, это войти в систему, введя свои имя и пароль. Теперь собственно просмотр процессов:

```
ps
```

Для просмотра процессов всех пользователей:

```
ps -A
```

Мониторинг и убийство процессов в Ubuntu

Способ второй - в консоли.

Для поиска программы в списке, для примера "skype":

```
ps -A | grep "skype"
```

Во всех этих списках вас должен интересовать только первый столбец с цифрами - это идентификационный номер процесса. Передав его команде "kill" можно завершить процесс:

```
sudo kill 1111
```

где "1111" - ID процесса.

Мониторинг и убийство процессов в Ubuntu

Способ второй - в консоли.

Утилиту kill можно вызвать с параметром "-9", в этом случае она не будет передавать процессу запрос "завершитесь, пожалуйста", как в первом случае, а просто убьет его без запроса. Зачастую избавится от зависших процессов можно избавиться только так. Например:

```
sudo kill -9 1111
```

Кроме того, существует также утилита killall, которая убивает процесс не по номеру, а по его имени, но использование это утилиты будет уместным не всегда, т.к. она убивает ВСЕ процессы с указанным именем (а их может быть несколько). Однако для нашего примера со skype это будет хороший вариант:

```
sudo killall skype
```