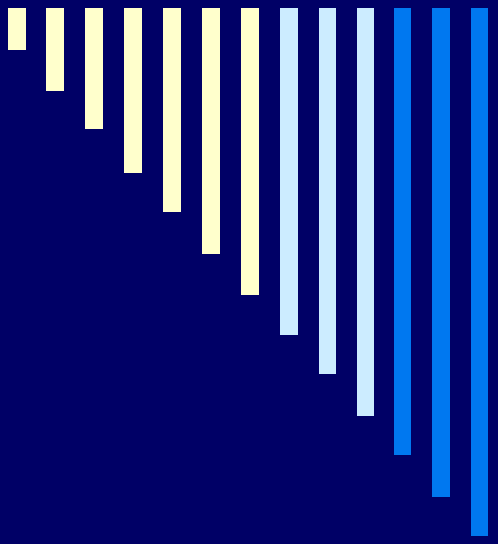


Системное программирование

курс дополнительных лекций

Лектор: Артамонов Евгений Борисович,

доцент кафедры КСУ НАУ

Структура и основные компоненты вычислительной системы :

- Работая на машине, мы реально не работаем с микросхемами и “железом”, наша работа происходит с программным обеспечением (ПО), которое размещено на аппаратуре. Поэтому вводится понятие Вычислительной системы.
- *Вычислительная система — это программно-аппаратный комплекс, который предоставляет услуги пользователю.*

Структура и основные компоненты вычислительной системы :

- Структуру вычислительной системы можно представить в виде пирамиды

Аппаратные средства

Ресурсы ВС разделяются на два типа:

- не участвующие в управлении программой (объем винчестера и т.д.).
- участвующие в управлении программой (размер ячейки памяти, объем оперативной памяти, скорость выполнения команд).

Ресурсы второго типа называются физическими ресурсами аппаратуры.

Управление физическими устройствами

- Управление физическими устройствами осуществляют программы, ориентированные на аппаратуру, взаимодействующие с аппаратными структурами, знающие "язык" аппаратуры.

Управление логическими устройствами

- Этот уровень ориентирован на пользователя. Команды данного уровня не зависят от физических устройств, они обращены к предыдущему уровню. На базе этого уровня могут создаваться новые логические ресурсы.

Системы программирования.

- *Система программирования — это комплекс программ для поддержки всего технологического цикла разработки программного обеспечения.*

Прикладное программное обеспечение

- Прикладное программное обеспечение необходимо для решения задач из конкретных областей

Понятие операционной системы

- *Операционная система (ОС) — программа, обеспечивающая взаимодействие пользователя с ВС, а также управляющая ресурсами ВС (логическими и физическими).* К ОС мы будем относить второй и третий уровень нашей пирамиды.

Структура ЭВМ

- Основной функцией центрального процессора (ЦП) является обработка информации и взаимодействие с устройствами. Обмениваться данными ЦП может только с ОЗУ (Оперативно Запоминающее Устройство).
- В ОЗУ размещается выполняемая в данный момент программа. ОЗУ состоит из ячеек памяти. Каждая ячейка имеет свой уникальный адрес, и каждая разбита на два поля: поле внутрисистемной информации и машинное слово.

Структура ЭВМ

ЦП состоит из двух компонентов:

- Устройство Управления (УУ) принимает очередное слово из ОЗУ и разбирается — команда это или данные. Если это команда — то УУ выполняет ее, иначе передает АУ.
- Арифметическое Устройство (АУ) занимается исключительно вычислениями.

Вычислительная система

— это некоторое объединение аппаратных средств, средств управления аппаратурой (физическими ресурсами), средств управления логическими ресурсами, систем программирования и прикладного программного обеспечения.

Операционная система

- К операционной системе в разных источниках относят либо два уровня управления — логический и физический, либо три — логический, физический и система программирования. Мы будем считать операционной системой два уровня — логический и физический. Мы начали рассматривать основные свойства этой иерархии, которую объявили, и нарисовали достаточно простую и традиционную схему, или структуру, вычислительной машины.

Операционная система

- Центральный процессор (ЦП) — это процессорный элемент, т.е. устройство, которое перерабатывает информацию, оперативная память (Оперативное Запоминающее Устройство, ОЗУ) и устройства управления внешними устройствами (УУВУ). Мы определили основное качество оперативной памяти: именно, в оперативной памяти лежит исполняемая в данный момент программа, и процессор все последующие команды исполняемой программы берет из оперативной памяти.

Основная проблема вычислительной техники

- Это несоответствие в скоростях доступа и обработки информации различных компонентов вычислительной системы, ведь у каждого компонента есть своя предыстория

Регистры

- В процессоре имеются устройства, способные хранить некоторую информацию.
- К этим устройствам возможен доступ прямым или косвенным способом из программы, выполняемой на машине.
- Есть группа регистров, которые называются регистрами общего назначения, которые доступны из всех команд.

Регистры

- СКОРОСТЬ ДОСТУПА К РЕГИСТРАМ ОБЩЕГО НАЗНАЧЕНИЯ СОИЗМЕРИМА СО СКОРОСТЬЮ ОБРАБОТКИ ИНФОРМАЦИИ В ПРОЦЕССОРЕ

Это означает, что торможение на участке процессор-оперативная память сокращается

Специальные регистры

- Первая подгруппа — это регистры, отвечающие за состояние исполняемой программы.
  - счетчик команд. Этот регистр содержит адрес исполняемой в данный момент команды.
  - регистр результата (flags), содержащий флаги результата выполнения последней команды.
  - регистр указателя стека. Есть команды, которые работают со стеком. Эти команды обычно используются для программирования переходов из функции и в функцию.
  - Автоматическая память занимается относительно вершины стека при входе в функцию, и, при выходе, она освобождается. Поэтому в автоматических переменных нельзя хранить данные после выхода из функции.

Специальные регистры

- Вторая подгруппа регистров — это регистры управления компонентами вычислительной системы, или управляющие регистры. Эти регистры связываются с УУВУ, и через эти регистры процессор может организовывать управление внешними устройствами.

Регистр управления жестким диском

- У него могут быть следующие поля:
  - Поле, указывающее, кому предназначена информация на этом регистре в данный момент времени (процессору или диску).
  - Если эта команда имеет формат “от процессора к устройству”, в нем может находиться код операции управления устройством, могут находиться некоторые операнды и т.д. Устройство пытается выполнить эту команду, и по результату ее выполнения возвращается результат так же в управляющий регистр.

Система прерываний

- К средствам, управляющим взаимосвязью с внешними устройствами, можно отнести систему прерываний.
- В каждой вычислительной машине имеется предопределенный, заданный при разработке и производстве, набор некоторых событий и аппаратных реакций на возникновение каждого из этих событий. Эти события называются прерываниями.
- Аппарат прерываний используется для управления внешними устройствами и для получения возможности асинхронной работы с внешними устройствами

Система прерываний

- Синхронная работа осуществляется так — система говорит “Дай мне блок информации”, а затем стоит и ждет этого блока.
- Работа асинхронная, если система говорит “Принеси мне, пожалуйста, блок информации” и продолжает свою работу, а когда приходит блок, она прерывается (по прерыванию завершения обмена) и принимает информацию. Такова схема прерываний.

Система прерываний - действия в аппаратуре ВС

- 1. Малое упрятывание: в некоторые специальные регистры аппаратно заносится (сохраняется) информация о выполняемой в данный момент программе. Это минимальные действия, необходимые для начала обработки прерывания. (*Обычно, в этот набор данных входит счетчик команд, регистр результата, указатель стека и несколько регистров общего назначения)*

Система прерываний - действия в аппаратуре ВС

- 1. Малое упрятывание
- 2. В некоторый специальный управляющий регистр, условно будем называть его регистром прерываний, помещается код возникшего прерывания.
- 3. Запускается программа обработки прерываний операционной системы.

Система прерываний - действия в аппаратуре ВС

- Запущенная программа в начале потребляет столько ресурсов (не более), сколько освобождено при аппаратном упрятывании информации. Эта программа производит анализ причины прерывания. Если это прерывание было фатальным (деление на ноль, например), то продолжать выполнение программы бессмысленно, и управление передается части операционной системы, которая эту программу выкинет.

Система прерываний

- Прерывания могут быть инициированы схемами контроля процессора (например, при делении на ноль), могут быть инициированы внешним устройством (при нажатии клавиши на клавиатуре возникает прерывание, по которому процессор считывает из некоторого регистра нажатый символ).

Регистры буферной памяти (Cache, КЭШ).

- Следующая группа регистров — регистры, относящиеся к т.н. буферной памяти. Мы возвращаемся к проблеме взаимодействия процессора и оперативной памяти и сглаживанию скоростей доступа в оперативную память.

Алгоритм чтения из оперативной памяти следующий

- Проверяется наличие в специальном регистровом буфере строчки, в которой находится исполнительный адрес, совпадающий с исполнительным адресом требуемого операнда. Если такая строчка имеется, то соответствующее этому адресу значение, считается значением операнда и передается в процессор для обработки (т.е. обращение в оперативную память не происходит).

Алгоритм чтения из оперативной памяти следующий

- Если такой строчки нет, то происходит обмен с оперативной памятью, и копия полученного значения помещается в регистровый буфер и помечается исполнительным адресом этого значения в оперативной памяти. Содержимое операнда поступает в процессор для обработки.

Алгоритм чтения из оперативной памяти следующий

- Аппаратно ищется свободная строка, и если таковая не найдена, запускается аппаратный процесс вытеснения из этого буфера наиболее “старой” строчки.

Алгоритм чтения из оперативной памяти следующий

- Данный алгоритм симметричен. Когда в программе встречается команда записи операнда в память, аппаратура выполняет следующие действия. Проверяется наличие в буфере строки с заданным исполнительным адресом. Если такая строка есть, то в поле “Содержимое” записывается новое значение и аппаратно корректируется признак старения строк. Если такой строчки нет, то запускается процесс выталкивания, и информация размещается в освободившейся строке.

Алгоритм чтения из оперативной памяти следующий

- Этот буфер чтения/записи служит достаточно мощным средством для минимизации обращений к ОЗУ. Наибольший эффект достигается при небольших циклах, когда все операнды размещаются в буфере, и после этого циклический процесс работает без обращений к ОЗУ. Иногда эти буфера называют КЭШ-буферами, а также ассоциативной памятью, потому что доступ к этой памяти осуществляется не по адресу (как в ОЗУ), а по значению поля.

- *Реально, все механизмы могут быть устроены иначе, чем мы здесь изучаем, т.к. мы изучаем некоторую обобщенную систему.*

Оперативная память

- Следующим компонентом, который мы с вами рассмотрим, с точки зрения системного подхода (а системный подход подразумевает то, что мы рассматриваем вещь не саму по себе, а в контексте взаимосвязи с другими компонентами) будут некоторые свойства ОЗУ.

1-й блок		2-й блок	. . .	k-й блок
0		1	. . .	k-1
k		k+1	. . .	2k-1
. . .		. . .	. . .	. . .

Использование расслоения памяти

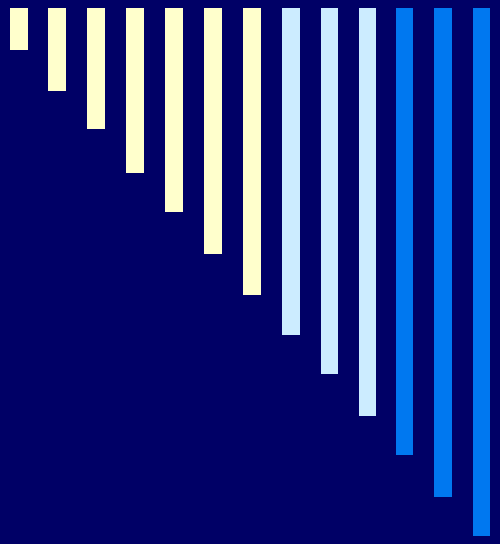
- Физически ОЗУ представимо в виде объединения k устройств, способных хранить одинаковое количество информации и способных взаимодействовать с процессором независимо друг от друга. При этом адресное пространство ВС организовано таким образом, что подряд идущие адреса, или ячейки памяти, находятся в соседних устройствах (блоках) оперативной памяти.

Использование расслоения памяти

- *Программа состоит (в большей степени) из линейных участков. Если использовать этот параллелизм, то можно организовать в процессоре еще один буфер, который организован так же, но в котором размещаются машинные команды. За счет того, что есть параллельно работающие устройства, то этот буфер автоматически заполняется вперед. Т.е. за одно обращение можно прочесть k машинных слов и разместить их в этом буфере.*

Использование расслоения памяти

- *Далее, действия с буфером команд похожи на действия с буфером чтения/записи. Когда нужна очередная команда (ее адрес находится в счетчике команд), происходит ее поиск (по адресу) в буфере, и если такая команда есть, то она считывается. Если такой команды нет, то опять-таки работает внутренний алгоритм выталкивания строки, новая строка считывается из памяти и копируется в буфер команд. Расслоение памяти в идеале увеличивает скорость доступа в k раз, плюс буфер команд позволяет сократить обращения к ОЗУ.*



Принципы работы операционных систем

Основные понятия операционной системы

- Идея в том, что ОС прежде всего система, обеспечивающая удобный интерфейс пользователям, соответствует рассмотрению сверху вниз.
- Другой взгляд, снизу вверх, дает представление об ОС как о некотором механизме, управляющем всеми частями сложной системы.

Основные понятия операционной системы

- Современные вычислительные системы состоят из процессоров, памяти, таймеров, дисков, накопителей на магнитных лентах, сетевых коммуникационной аппаратуры, принтеров и других устройств.

- В соответствии со вторым подходом функцией ОС является распределение процессоров, памяти, устройств и данных между процессами, конкурирующими за эти ресурсы.

Основные понятия операционной системы

- ОС должна управлять всеми ресурсами вычислительной машины таким образом, чтобы обеспечить максимальную эффективность ее функционирования. Критерием эффективности может быть, например, пропускная способность или реактивность системы.

Основные понятия операционной системы

Управление ресурсами включает решение двух общих, не зависящих от типа ресурса задач:

- планирование ресурса - то есть определение, кому, когда, а для делимых ресурсов и в каком количестве, необходимо выделить данный ресурс;
- отслеживание состояния ресурса - то есть поддержание оперативной информации о том занятости ресурса, а для делимых ресурсов - какое количество ресурса уже распределено, а какое свободно.

Классификация операционных систем

- Операционные системы могут различаться особенностями реализации внутренних алгоритмов управления основными ресурсами компьютера (процессорами, памятью, устройствами), особенностями использованных методов проектирования, типами аппаратных платформ, областями использования и многими другими свойствами.

Классификация операционных систем

- **Особенности алгоритмов управления ресурсами.** От эффективности алгоритмов управления локальными ресурсами компьютера во многом зависит эффективность всей сетевой ОС в целом. Поэтому, характеризуя сетевую ОС, часто приводят важнейшие особенности реализации функций ОС по управлению процессорами, памятью, внешними устройствами автономного компьютера.

Классификация операционных систем

- В зависимости от особенностей использованного алгоритма управления процессором, операционные системы делят на многозадачные и однозадачные, многопользовательские и однопользовательские, на системы, поддерживающие многонитевую обработку и не поддерживающие ее, на многопроцессорные и однопроцессорные системы.

Классификация операционных систем

Поддержка многозадачности. По числу одновременно выполняемых задач операционные системы могут быть разделены на два класса:

- однозадачные (например, MS-DOS, MSX);
- многозадачные (ОС ЕС, OS/2, UNIX, Windows).

Классификация операционных систем

Однозадачные ОС в основном выполняют функцию предоставления пользователю виртуальной машины, делая более простым и удобным процесс взаимодействия пользователя с компьютером. Однозадачные ОС включают средства управления периферийными устройствами, средства управления файлами, средства общения с пользователем. Многозадачные ОС, кроме вышеперечисленных функций, управляют разделением совместно используемых ресурсов, таких как процессор, оперативная память, файлы и внешние устройства.

Классификация операционных систем

Поддержка многопользовательского режима. По числу одновременно работающих пользователей ОС делятся на:

- Однопользовательские – “устаревшие” (MS-DOS, Windows 3.x, ранние версии OS/2);
- многопользовательские (UNIX, Windows NT, Windows XP и т.д.).

Классификация операционных систем

Главным отличием многопользовательских систем от однопользовательских является наличие средств защиты информации каждого пользователя от несанкционированного доступа других пользователей. Следует заметить, что не всякая многозадачная система является многопользовательской, и не всякая однопользовательская ОС является однозадачной.

Классификация операционных систем

Вытесняющая и невытесняющая многозадачность. Важнейшим разделяемым ресурсом является процессорное время. Способ распределения процессорного времени между несколькими одновременно существующими в системе процессами (или нитями) во многом определяет специфику ОС.

Классификация операционных систем

Среди множества существующих вариантов реализации многозадачности можно выделить две группы алгоритмов:

- невытесняющая многозадачность (NetWare, Windows 3.x);
- вытесняющая многозадачность (Windows выше NT, OS/2, UNIX).

Классификация операционных систем

Поддержка многонитевости. Важным свойством операционных систем является возможность распараллеливания вычислений в рамках одной задачи. Многонитевая ОС разделяет процессорное время не между задачами, а между их отдельными ветвями (нитями).

Классификация операционных систем

- Многопроцессорная обработка. Другим важным свойством ОС является отсутствие или наличие в ней средств поддержки многопроцессорной обработки - мультипроцессирование. Мультипроцессирование приводит к усложнению всех алгоритмов управления ресурсами. В наши дни становится общепринятым введение в ОС функций поддержки многопроцессорной обработки данных.
- Многопроцессорные ОС могут классифицироваться по способу организации вычислительного процесса в системе с многопроцессорной архитектурой: асимметричные ОС и симметричные ОС.

Классификация операционных систем

- Асимметричная ОС целиком выполняется только на одном из процессоров системы, распределяя прикладные задачи по остальным процессорам.
- Симметричная ОС полностью децентрализована и использует весь пул процессоров, разделяя их между системными и прикладными задачами.

Классификация операционных систем

Специфика ОС проявляется и в том, каким образом она реализует сетевые функции: распознавание и перенаправление в сеть запросов к удаленным ресурсам, передача сообщений по сети, выполнение удаленных запросов. При реализации сетевых функций возникает комплекс задач, связанных с распределенным характером хранения и обработки данных в сети: ведение справочной информации о всех доступных в сети ресурсах и серверах, адресация взаимодействующих процессов, обеспечение прозрачности доступа, тиражирование данных, согласование копий, поддержка безопасности данных.

Особенности областей использования операционных систем

Многозадачные ОС подразделяются на три типа в соответствии с использованными при их разработке критериями эффективности:

- системы пакетной обработки (например, ОС ЕС);
- системы разделения времени (UNIX, VMS);
- системы реального времени (QNX, RT/11).

Особенности областей использования операционных систем

- **СИСТЕМЫ ПАКЕТНОЙ ОБРАБОТКИ предназначались для решения задач в основном вычислительного характера, не требующих быстрого получения результатов. Главной целью и критерием эффективности систем пакетной обработки является максимальная пропускная способность, то есть решение максимального числа задач в единицу времени.**
- **В таких ОС невозможно гарантировать выполнение того или иного задания в течение определенного периода времени. В системах пакетной обработки переключение процессора с выполнения одной задачи на выполнение другой происходит только в случае, если активная задача сама отказывается от процессора, например, из-за необходимости выполнить операцию ввода-вывода. Поэтому одна задача может надолго занять процессор, что делает невозможным выполнение интерактивных задач.**

Особенности областей использования операционных систем

- **СИСТЕМЫ РАЗДЕЛЕНИЯ ВРЕМЕНИ призваны исправить основной недостаток систем пакетной обработки - изоляцию пользователя-программиста от процесса выполнения его задач. Каждому пользователю системы разделения времени предоставляется терминал, с которого он может вести диалог со своей программой.**
- **Системы разделения времени обладают меньшей пропускной способностью, чем системы пакетной обработки, так как на выполнение принимается каждая запущенная пользователем задача, а не та, которая "выгодна" системе, и, кроме того, имеются накладные расходы вычислительной мощности на более частое переключение процессора с задачи на задачу. Критерием эффективности систем разделения времени является не максимальная пропускная способность, а удобство и эффективность работы пользователя.**

Особенности областей использования операционных систем

- Системы реального времени применяются для управления различными техническими объектами, такими, например, как станок, спутник, научная экспериментальная установка или технологическими процессами.
- Во всех этих случаях существует предельно допустимое время, в течение которого должна быть выполнена та или иная программа, управляющая объектом.
- Критерием эффективности для систем реального времени является их способность выдерживать заранее заданные интервалы времени между запуском программы и получением результата (управляющего воздействия). Это время называется временем реакции системы, а соответствующее свойство системы - реактивностью.

Особенности областей использования операционных систем

- Некоторые операционные системы могут совмещать в себе свойства систем разных типов, например, часть задач может выполняться в режиме пакетной обработки, а часть - в режиме реального времени или в режиме разделения времени. В таких случаях режим пакетной обработки часто называют фоновым режимом.

Управление процессами

- Важнейшей частью операционной системы, непосредственно влияющей на функционирование вычислительной машины, является подсистема управления процессами.
- Процесс (или по-другому, задача) - абстракция, описывающая выполняющуюся программу. Для операционной системы процесс представляет собой единицу работы, заявку на потребление системных ресурсов.

Управление процессами

- Подсистема управления процессами планирует выполнение процессов, то есть распределяет процессорное время между несколькими одновременно существующими в системе процессами, а также занимается созданием и уничтожением процессов, обеспечивает процессы необходимыми системными ресурсами, поддерживает взаимодействие между процессами.

Состояние процессов

В многозадачной (многопроцессной) системе процесс может находиться в одном из трех основных состояний:

- ВЫПОЛНЕНИЕ - активное состояние процесса, во время которого процесс обладает всеми необходимыми ресурсами и непосредственно выполняется процессором;

Состояние процессов

В многозадачной (многопроцессной) системе процесс может находиться в одном из трех основных состояний:

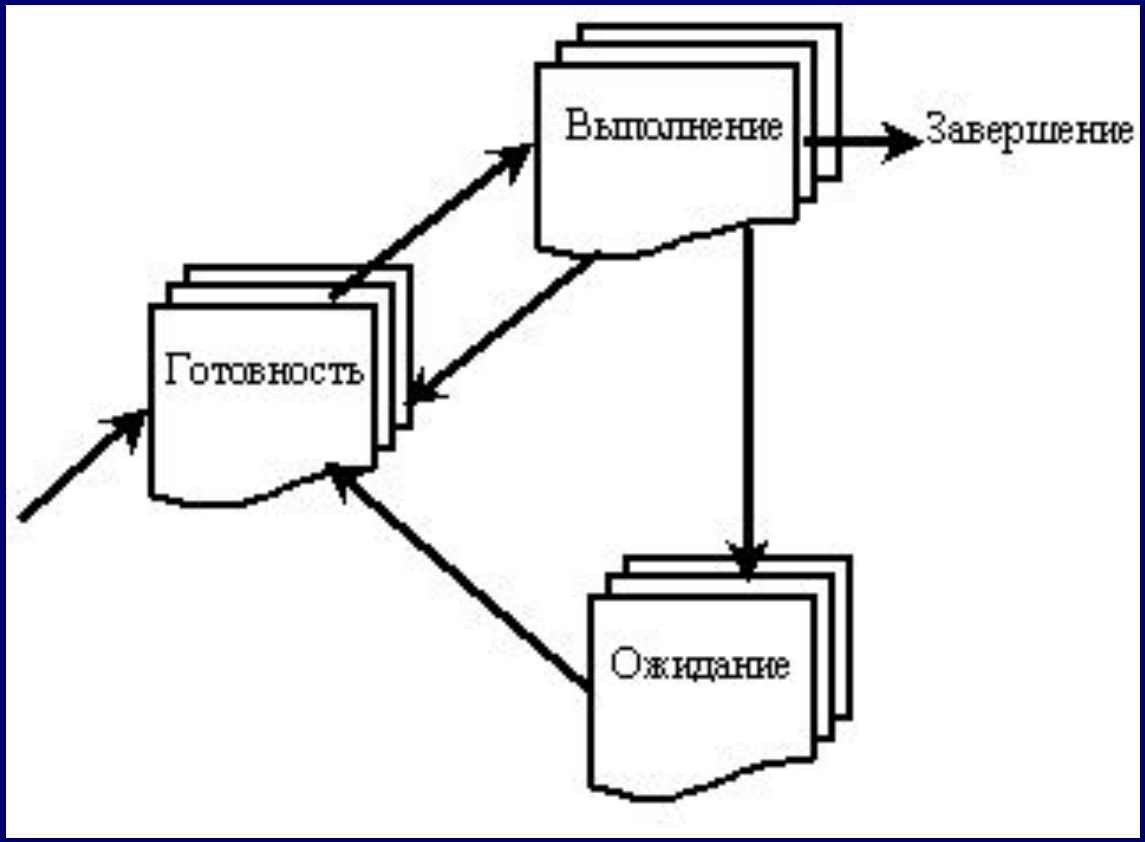
- ВЫПОЛНЕНИЕ;
- ОЖИДАНИЕ - пассивное состояние процесса, процесс заблокирован, он не может выполняться по своим внутренним причинам, он ждет осуществления некоторого события;

Состояние процессов

В многозадачной (многопроцессной) системе процесс может находиться в одном из трех основных состояний:

- ВЫПОЛНЕНИЕ;
- ОЖИДАНИЕ;
- ГОТОВНОСТЬ - также пассивное состояние процесса - но процесс заблокирован в связи с внешними по отношению к нему обстоятельствами: процесс имеет все требуемые для него ресурсы, он готов выполняться, однако процессор занят выполнением другого процесса.

Граф состояний процесса в многозадачной среде



Алгоритмы планирования процессов

Планирование процессов включает в себя решение следующих задач:

- определение момента времени для смены выполняемого процесса;
- выбор процесса на выполнение из очереди готовых процессов;
- переключение контекстов "старого" и "нового" процессов.

Алгоритмы планирования процессов

Первые две задачи решаются программными средствами, а последняя в значительной степени аппаратно. Среди множества алгоритмов рассмотрим подробнее две группы наиболее часто встречающихся алгоритмов: алгоритмы, основанные на квантовании, и алгоритмы, основанные на приоритетах.

Алгоритмы планирования процессов

В соответствии с алгоритмами, основанными на квантовании, смена активного процесса происходит, если:

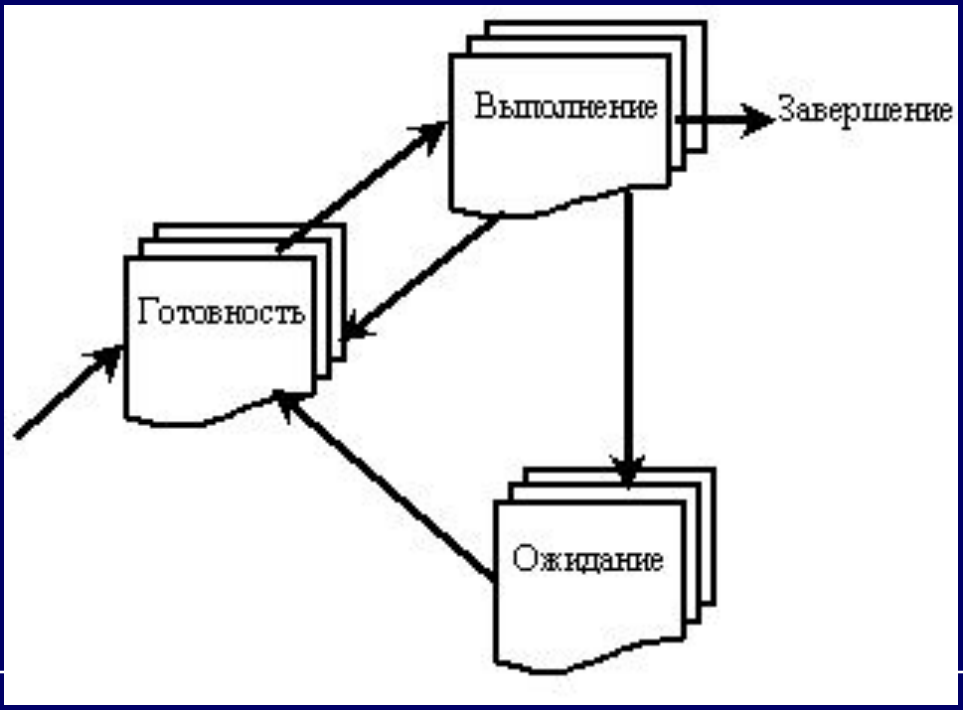
- процесс завершился и покинул систему,
- произошла ошибка,
- процесс перешел в состояние ОЖИДАНИЕ,
- исчерпан квант процессорного времени, отведенный данному процессу.

Алгоритмы планирования процессов

Процесс, который исчерпал свой квант, переводится в состояние ГОТОВНОСТЬ и ожидает, когда ему будет предоставлен новый квант процессорного времени, а на выполнение в соответствии с определенным правилом выбирается новый процесс из очереди готовых. Таким образом, ни один процесс не занимает процессор надолго, поэтому квантование широко используется в системах разделения времени.

Алгоритмы планирования процессов

Граф состояний процесса, изображенный на рисунке соответствует алгоритму планирования, основанному на квантовании



Алгоритмы планирования процессов

Кванты, выделяемые процессам, могут быть одинаковыми для всех процессов или различными. Кванты, выделяемые одному процессу, могут быть фиксированной величины или изменяться в разные периоды жизни процесса.

Алгоритмы планирования процессов

Процессы, которые не полностью использовали выделенный им квант (например, из-за ухода на выполнение операций ввода-вывода), могут получить или не получить компенсацию в виде привилегий при последующем обслуживании. По разному может быть организована очередь готовых процессов: циклически, по правилу "первый пришел - первый обслужился" (FIFO) или по правилу "последний пришел - первый обслужился" (LIFO).

Алгоритмы планирования процессов

Другая группа алгоритмов использует понятие "приоритет" процесса. Приоритет - это число, характеризующее степень привилегированности процесса при использовании ресурсов вычислительной машины, в частности, процессорного времени: чем выше приоритет, тем выше привилегии. Приоритет может выражаться целыми или дробными, положительным или отрицательным значением. Чем выше привилегии процесса, тем меньше времени он будет проводить в очередях.

Приоритетные алгоритмы

Существует две разновидности приоритетных алгоритмов:

- алгоритмы, использующие относительные приоритеты,
- алгоритмы, использующие абсолютные приоритеты.

Приоритетные алгоритмы

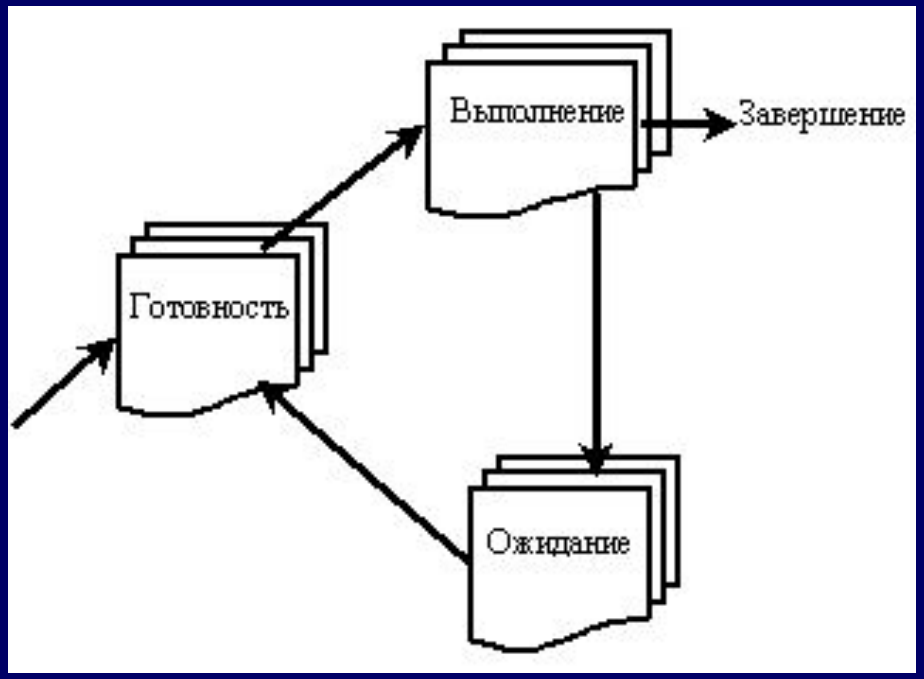
В обоих случаях выбор процесса на выполнение из очереди готовых осуществляется одинаково: выбирается процесс, имеющий наивысший приоритет. По разному решается проблема определения момента смены активного процесса.

Приоритетные алгоритмы

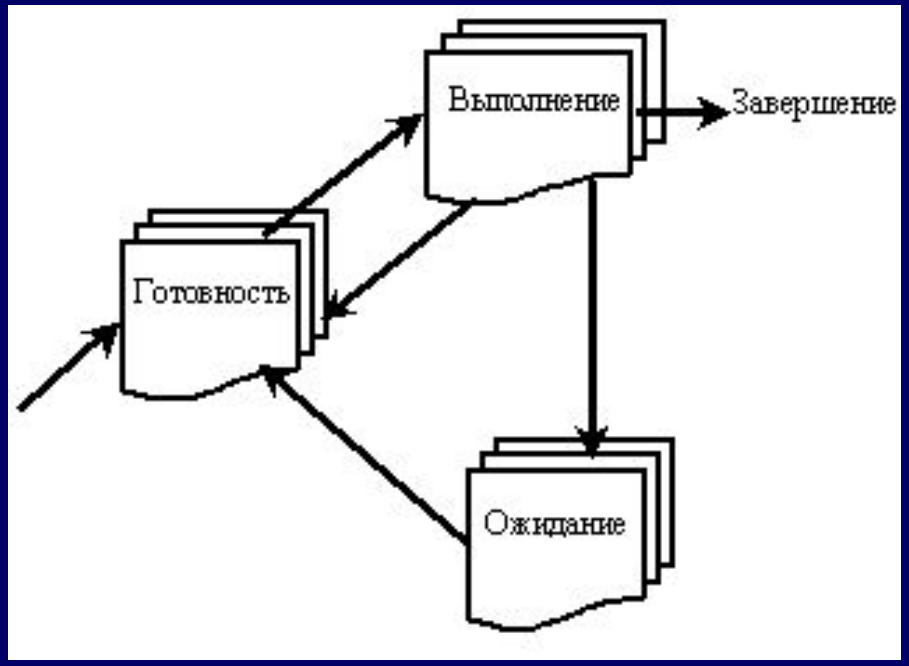
- В системах с относительными приоритетами активный процесс выполняется до тех пор, пока он сам не покинет процессор, перейдя в состояние ОЖИДАНИЕ (или же произойдет ошибка, или процесс завершится).
- В системах с абсолютными приоритетами выполнение активного процесса прерывается еще при одном условии: если в очереди готовых процессов появился процесс, приоритет которого выше приоритета активного процесса. В этом случае прерванный процесс переходит в состояние готовности.

Приоритетные алгоритмы

графы состояний процесса для алгоритмов



с относительными приоритетами



с абсолютными приоритетами

Вытесняющие и невытесняющие алгоритмы планирования

- Non-preemptive multitasking - невытесняющая многозадачность - это способ планирования процессов, при котором активный процесс выполняется до тех пор, пока он сам, по собственной инициативе, не отдаст управление планировщику операционной системы для того, чтобы тот выбрал из очереди другой, готовый к выполнению процесс.

Вытесняющие и невытесняющие алгоритмы планирования

- Preemptive multitasking - вытесняющая многозадачность - это такой способ, при котором решение о переключении процессора с выполнения одного процесса на выполнение другого процесса принимается планировщиком операционной системы, а не самой активной задачей.

Вытесняющие и невытесняющие алгоритмы планирования

- Существенным преимуществом non-preemptive систем является более высокая скорость переключения с задачи на задачу. Примером эффективного использования невытесняющей многозадачности является файл-сервер NetWare, в котором, в значительной степени благодаря этому, достигнута высокая скорость выполнения файловых операций. Менее удачным оказалось использование невытесняющей многозадачности в операционной среде Windows 3.x.

Вытесняющие и невытесняющие алгоритмы планирования

- Однако почти во всех современных операционных системах, ориентированных на высокопроизводительное выполнение приложений (UNIX, Windows XP, OS/2), реализована вытесняющая многозадачность. Иногда вытесняющую многозадачность часто называют истинной многозадачностью.

Функции планировщика-диспетчера

- Во многих ОС эти функции могут быть представлены неразрывной последовательностью, поэтому введем термин планировщик-диспетчер.

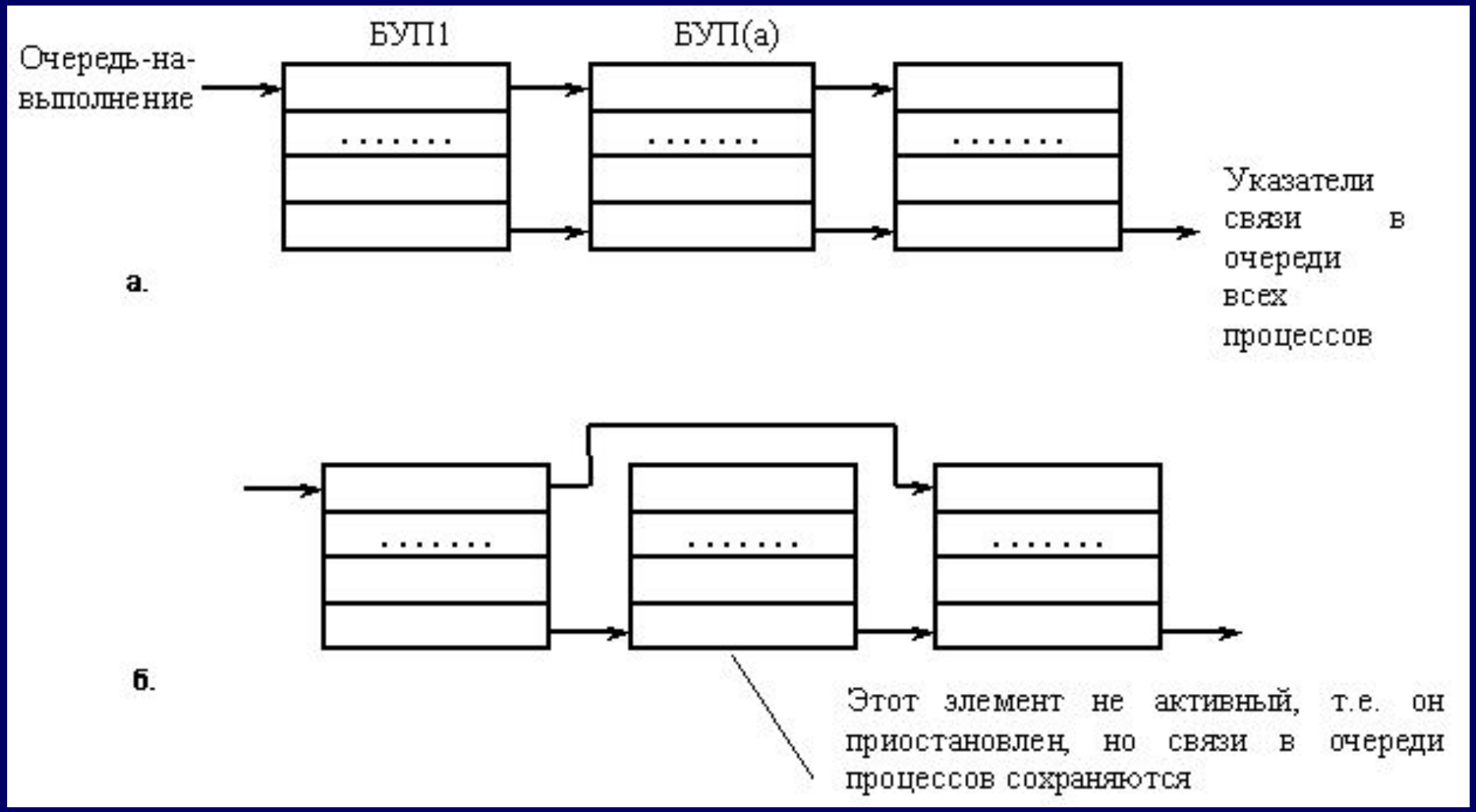
Функции планировщика-диспетчера

- Планировщик - это программа, ответственная за постановку процессов в очередь-на-выполнение и управляющая этой очередью.
- Диспетчер - это программа, которая выбирает процессы из очереди-на-выполнение, переводит их в активное состояние и передает им контроль над CPU.
- Основная функция - возможность управлять действиями большого числа процессов.

Планировщик-диспетчер – приостановка процесса

- В ходе своего выполнения системный процесс может установить, что требуемый ресурс занят или в данный момент процессу не требуется производить каких-либо действий.
- Тогда процесс может "сознательно"приостановить свое выполнение до момента активизации его другим процессом.
- При этой процедуре адрес блока управления процессом (БУП) записывается в стек, затем для активизации другого процесса вызывается системный планировщик.
- Результатом этого действия является то, что "приостановленный" процесс не помещается в очередь-на-выполнение, т.е. не активизируется.

Планировщик-диспетчер – приостановка процесса



Планировщик-диспетчер – приостановка процесса

- Алгоритм "Приостановить процесс"
- НАЧАЛО
  Анализ прерывания
  procedure "ПРИОСТ"
  IF1<приостанавлемый процесс не является системным и находится в ОП>
  блокировать процесс по условию выгрузки из ОП
  IF2<есть свободный блок запроса в очереди приостановок>
  создать блок запроса в очереди приостановок
  разблокировать процесс "откачки"
  формировать код возврата по удачному завершению передать управление диспетчеру
  FI2
  формирование кода возврата по перегрузке системы
  передать управление диспетчеру
  FI1формировать код возврата по невозможности приостановки
  передать управление диспетчеру
- КОНЕЦ

Планировщик-диспетчер – отсрочка процесса

- Алгоритм "Отсрочка процесса "
- НАЧАЛО

Анализ прерывания
procedure "ВОЗОБНОВИТЬ"
IF1<процесс не системный и был выгружен из ОП>
IF2<есть свободные блоки запросов в очереди возобновлений>
поместить блок запроса в очередь возобновлений
разблокировать процесс подкачек
блокировать процесс по условию загрузки в ОП
формировать код возврата по удачному завершению
переход к диспетчеру
FI2
формирование кода возврата по перегрузке системы
переход к диспетчеру
FI1
формировать код возврата по невозможности приостановки
переход к диспетчеру

- КОНЕЦ

Планировщик-диспетчер – активизация процесса

Алгоритм "Активизация процесса "

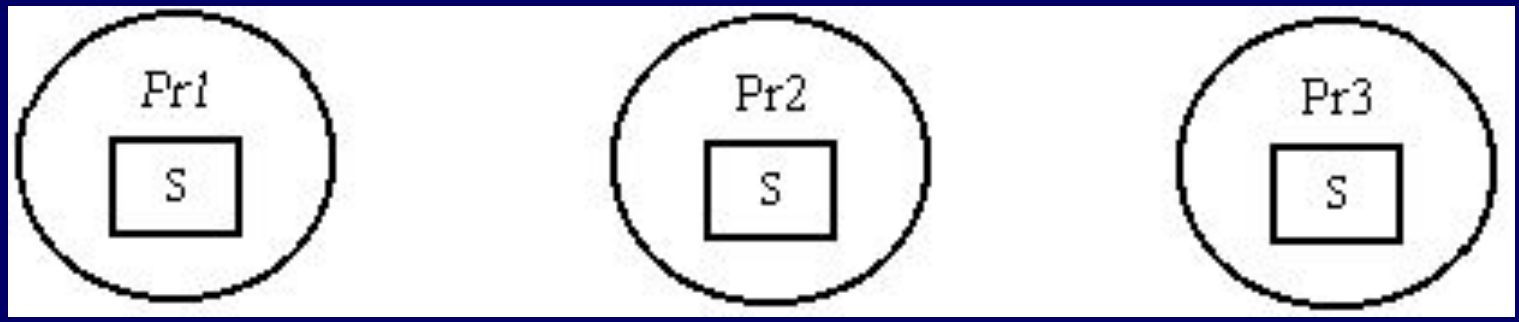
- Процедура "Активизировать" помещает БУП в очередь-на-выполнение.
- Проверяет, совпадают ли адрес БУП с адресом текущего выполняющегося процесса.
- Если не совпадают, то процесс помещается в очередь-на-выполнение.
- Если совпадает, то БУП уже находится в конце очереди-на-выполнение.

Планировщик-диспетчер

Планировщик-диспетчер может разделяться процессами, т.е. планировщик вызывается путем обращений i-го прцесса к данной системной программе S, что является косвенным результатом операции ядра ОС. Ядро и планировщик тогда потенциально содержатся в адресном пространстве всех процессов и выполняются в составе любого процесса.

Планировщик-диспетчер

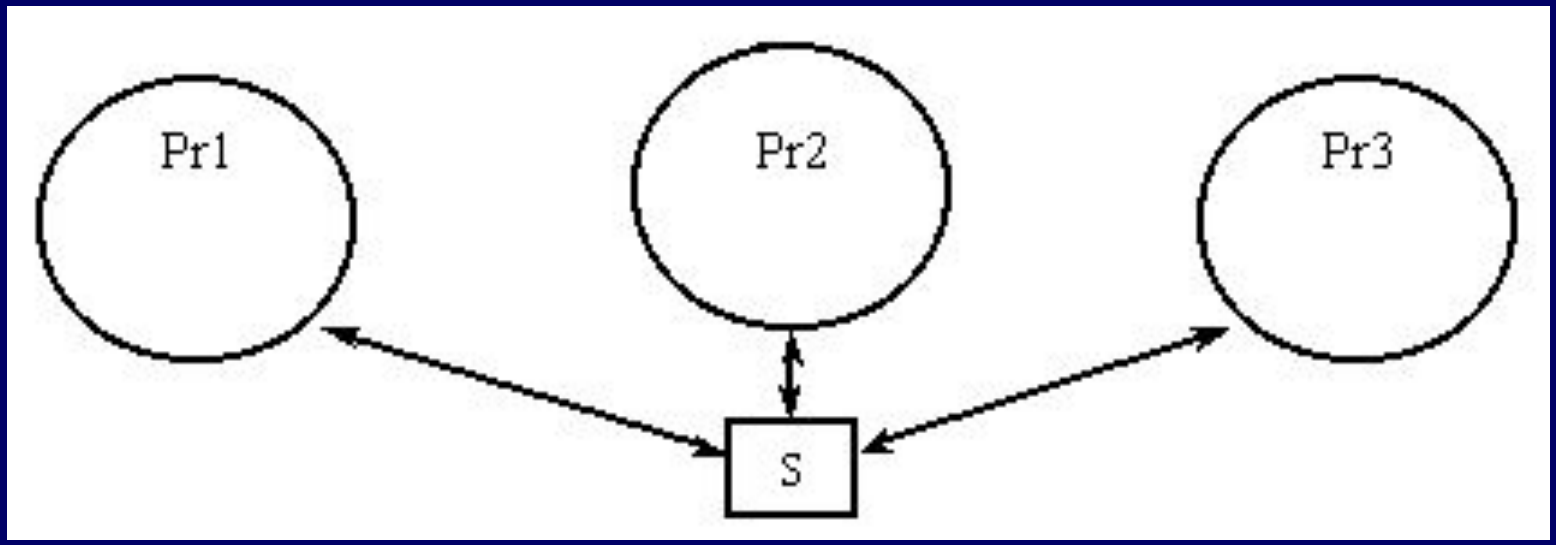
Разделение планировщика-диспетчера



- S - планировщик;
- Pri - процесс.

Планировщик-диспетчер

Централизация планировщика-диспетчера

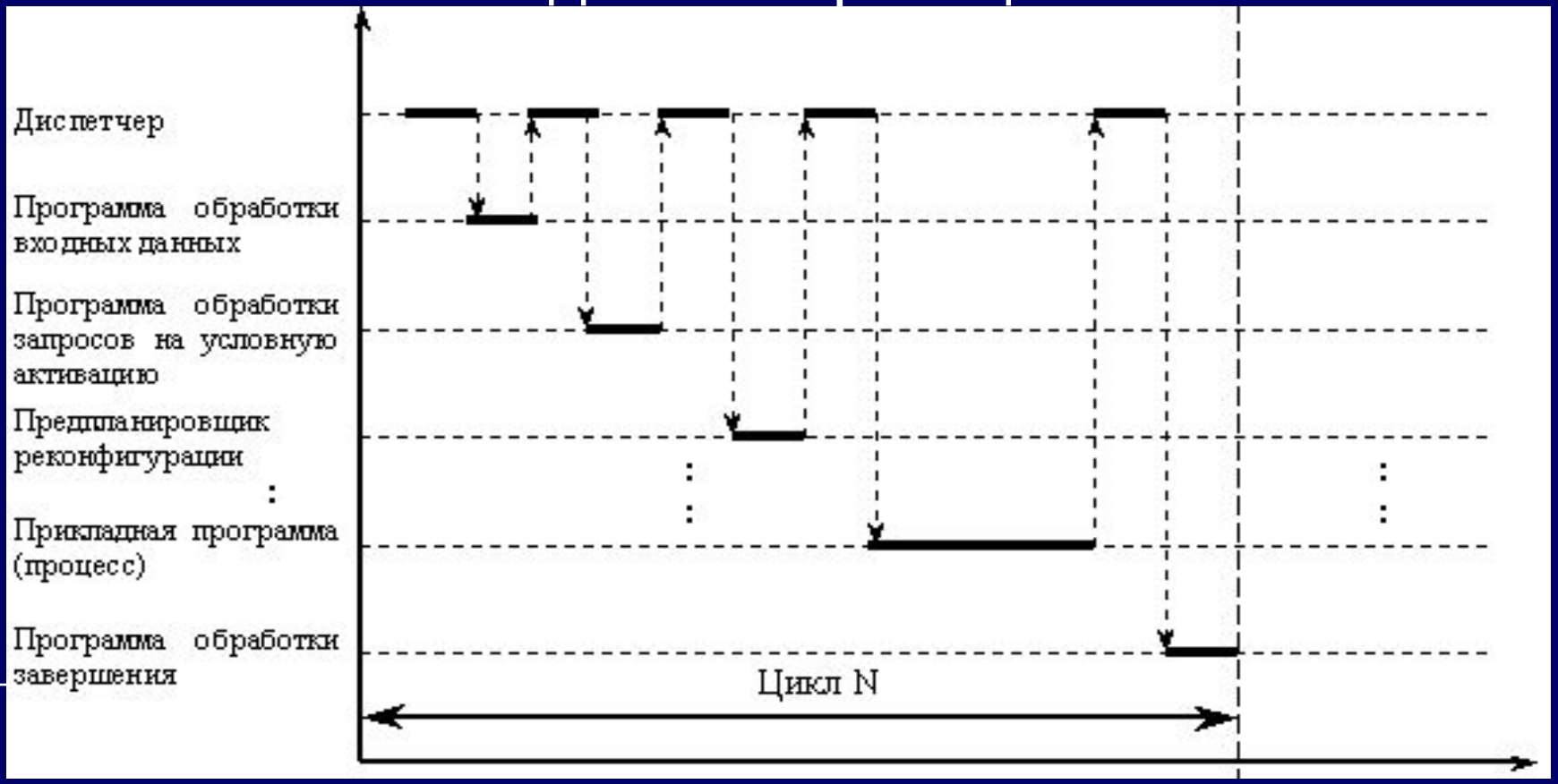


Планировщик-диспетчер

Этот тип планировщика считается отдельным процессом, он может непрерывно проверять заявки системы на планирование или может активизироваться соответствующими сигналами.

Планировщик-диспетчер

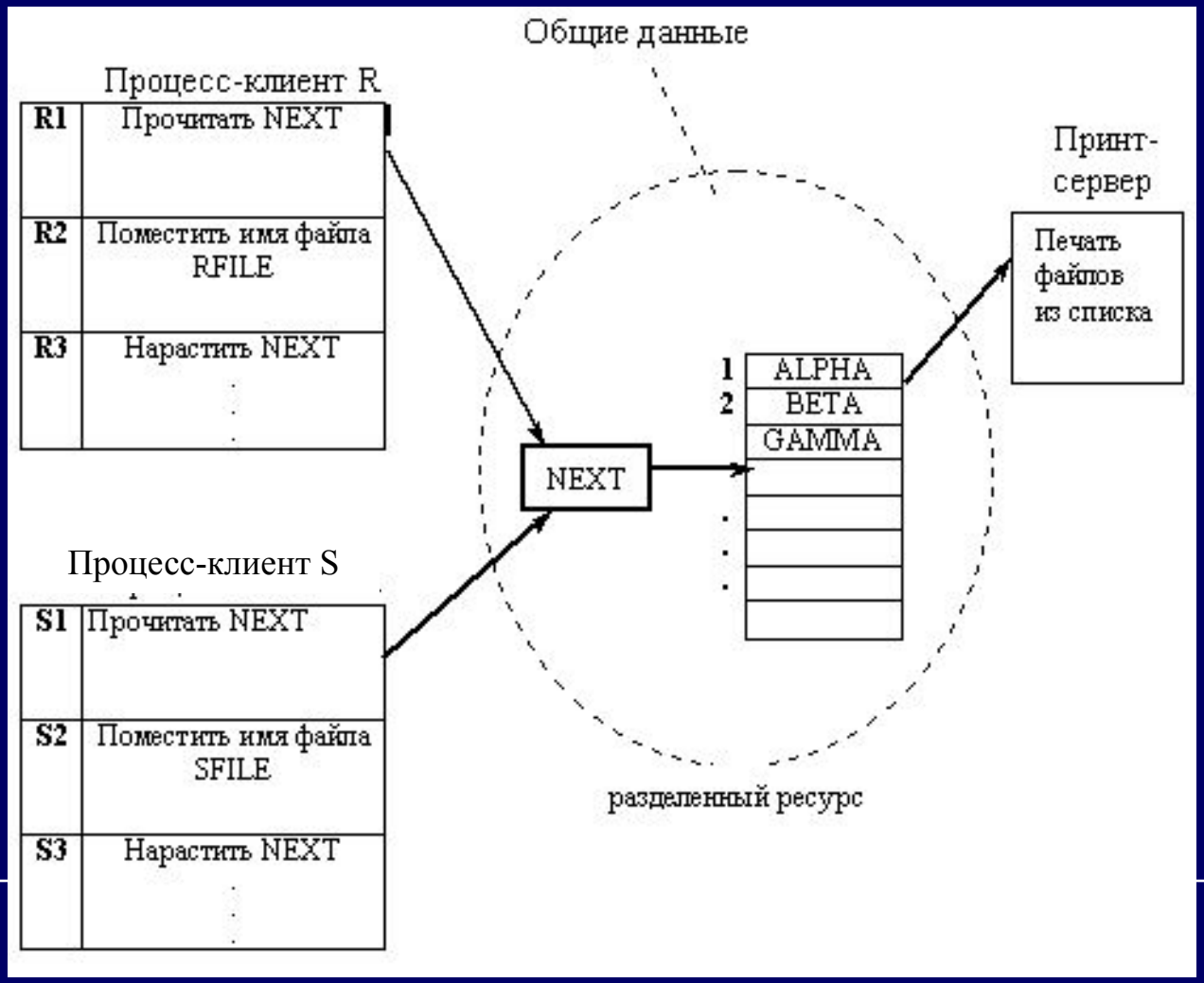
Графическая интерпретация планирования и диспетчеризации



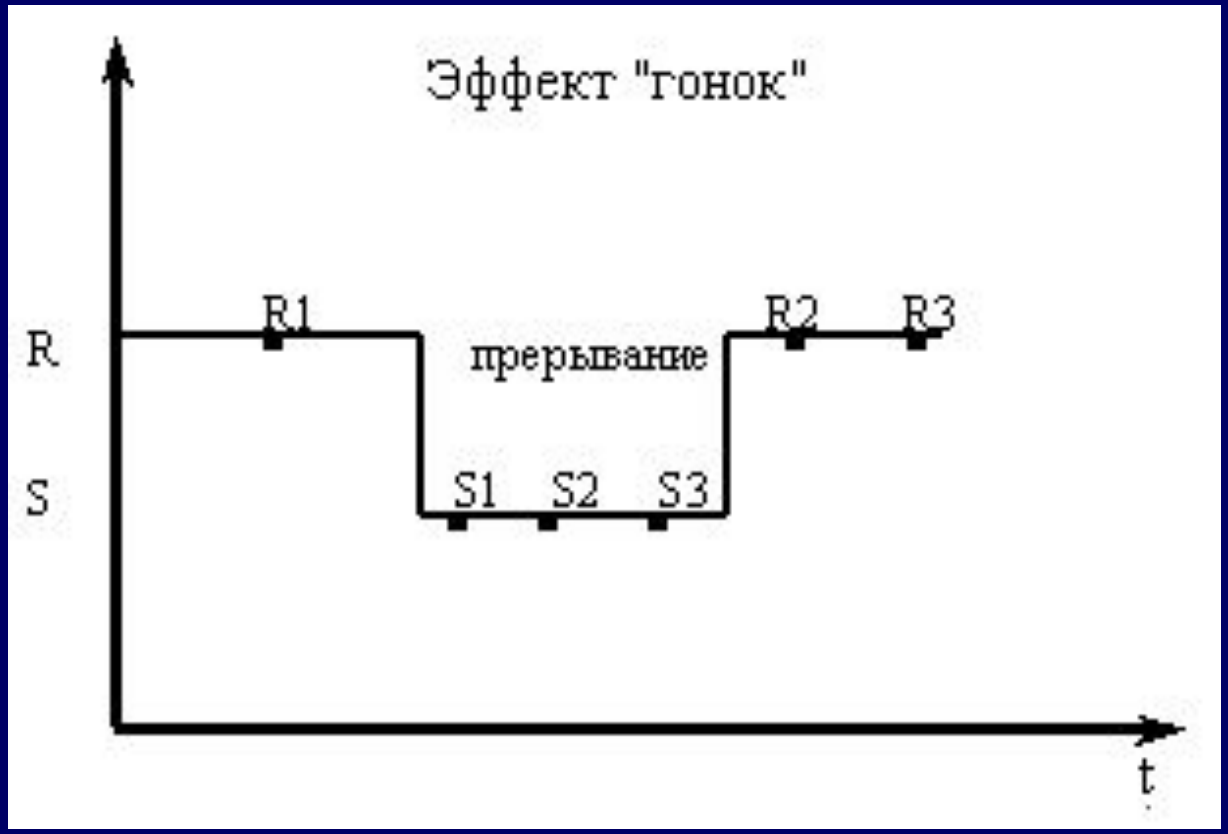
Средства синхронизации и взаимодействия процессов

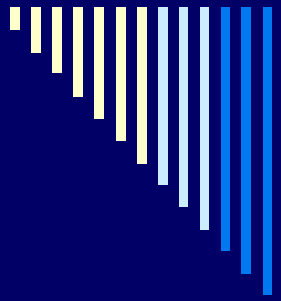
Процессам часто нужно взаимодействовать друг с другом, например, один процесс может передавать данные другому процессу, или несколько процессов могут обрабатывать данные из общего файла. Во всех этих случаях возникает проблема синхронизации процессов, которая может решаться приостановкой и активизацией процессов, организацией очередей, блокированием и освобождением ресурсов.

Средства синхронизации и взаимодействия процессов



Средства синхронизации и взаимодействия процессов





Проблема синхронизации

Пренебрежение вопросами синхронизации процессов, выполняющихся в режиме мультипрограммирования, может привести к их неправильной работе или даже к краху системы.

Рассмотрим программу печати файлов (принт-сервер).

Эта программа печатает по очереди все файлы, имена которых последовательно в порядке поступления записывают в специальный общедоступный файл "заказов" другие программы. Особая переменная NEXT, также доступная всем процессам-клиентам, содержит номер первой свободной для записи имени файла позиции файла "заказов".

Проблема синхронизации

Предположим, что в некоторый момент процесс R решил распечатать свой файл, для этого он прочитал значение переменной NEXT, значение которой для определенности предположим равным 4.

Процесс запомнил это значение, но поместить имя файла не успел, так как его выполнение было прервано (например, в следствие исчерпания кванта).

Очередной процесс S, желающий распечатать файл, прочитал то же самое значение переменной NEXT, поместил в четвертую позицию имя своего файла и нарастил значение переменной на единицу.

Проблема синхронизации

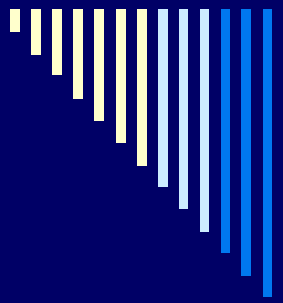
Когда в очередной раз управление будет передано процессу R, то он, продолжая свое выполнение, в полном соответствии со значением текущей свободной позиции, полученным во время предыдущей итерации, запишет имя файла также в позицию 4, поверх имени файла процесса S.

ТАКИМ ОБРАЗОМ, ПРОЦЕСС S НИКОГДА НЕ УВИДИТ СВОЙ ФАЙЛ РАСПЕЧАТАННЫМ.

Проблема синхронизации

Сложность проблемы синхронизации состоит в нерегулярности возникающих ситуаций.

Можно представить и другое развитие событий: были потеряны файлы нескольких процессов или, напротив, не был потерян ни один файл. В данном случае все определяется взаимными скоростями процессов и моментами их прерывания. Поэтому отладка взаимодействующих процессов является сложной задачей. Ситуации подобные той, когда два или более процессов обрабатывают разделяемые данные, и конечный результат зависит от соотношения скоростей процессов, называются гонками.



Критическая секция

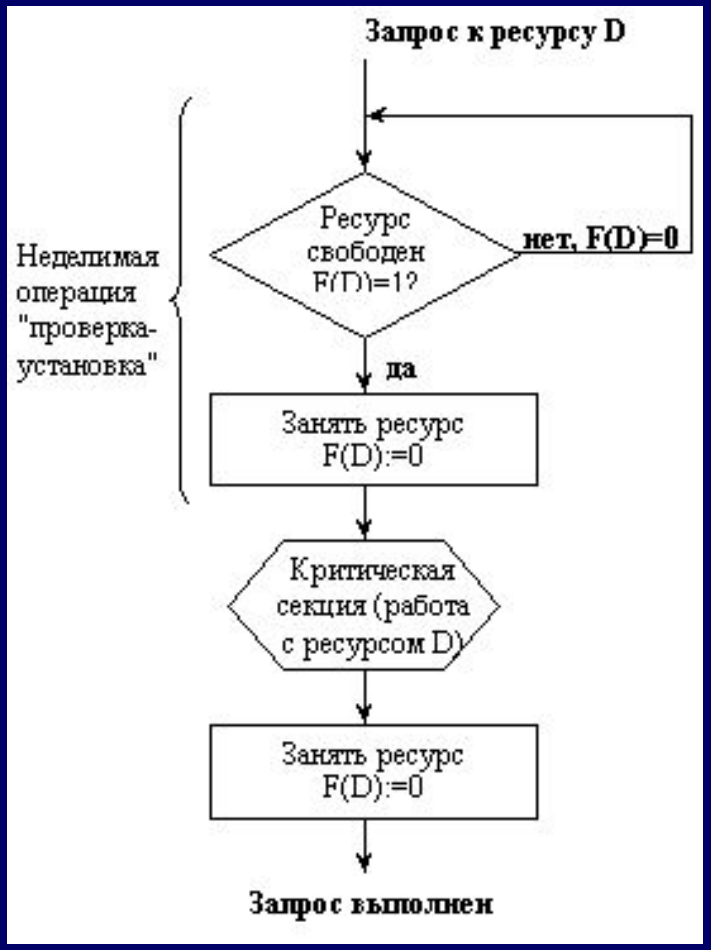
Важным понятием синхронизации процессов является понятие "критическая секция" программы (CS). Критическая секция - это часть программы, в которой осуществляется доступ к разделяемым данным. Чтобы исключить эффект гонок по отношению к некоторому ресурсу, необходимо обеспечить, чтобы в каждый момент в критической секции, связанной с этим ресурсом, находился максимум один процесс.

Критическая секция

Этот прием называют взаимным исключением. Простейший способ обеспечить взаимное исключение - позволить процессу, находящемуся в критической секции, запрещать все прерывания.

Этот способ непригоден, так как опасно доверять управление системой пользовательскому процессу; он может надолго занять процессор, а при крахе процесса в критической области крах потерпит вся система.

Критическая секция

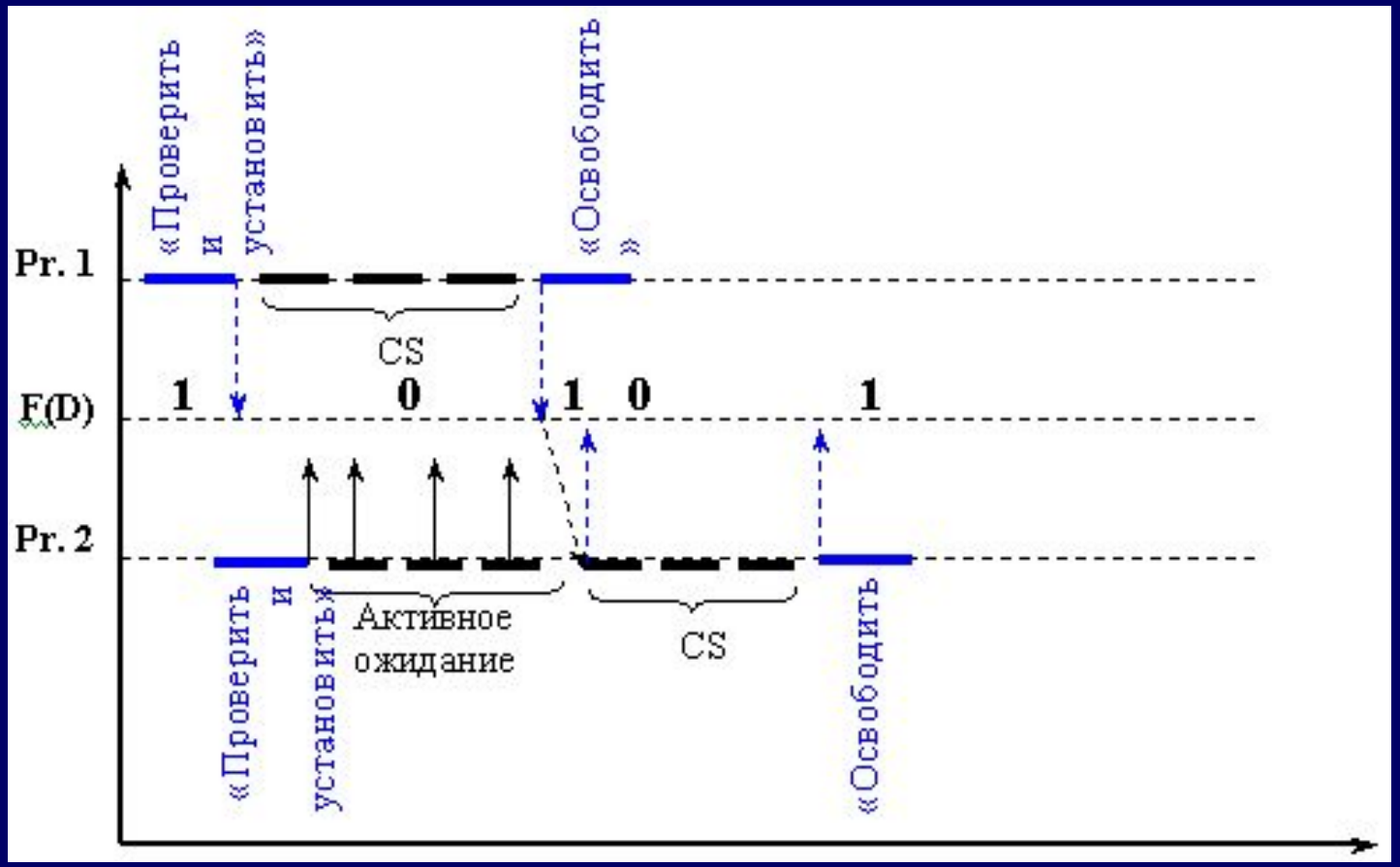


Реализация критических секций с использованием блокирующих переменных

Критическая секция

На рисунке показан фрагмент алгоритма процесса, использующего для реализации взаимного исключения доступа к разделяемому ресурсу D блокирующую переменную F(D). Перед входом в критическую секцию процесс проверяет, свободен ли ресурс D. Если он занят, то проверка циклически повторяется, если свободен, то значение переменной F(D) устанавливается в 0, и процесс входит в критическую секцию. После того, как процесс выполнит все действия с разделяемым ресурсом D, значение переменной F(D) снова устанавливается равным 1.

Критическая секция



Временная диаграмма исполнения команды "Проверить и установить"

Управление памятью

Память является важнейшим ресурсом, требующим тщательного управления со стороны мультипрограммной операционной системы. Распределению подлежит вся оперативная память, не занятая операционной системой.

Управление памятью

Функциями ОС по управлению памятью являются:

- отслеживание свободной и занятой памяти,
- выделение памяти процессам
- освобождение памяти при завершении процессов,
- вытеснение процессов из оперативной памяти на диск
- возвращение их в оперативную память
- настройка адресов программы на конкретную область физической памяти.

Управление памятью

Цель управления оперативной памятью:

- уменьшить пустые пространства памяти (т.е. фрагментацию), возникающие из-за того, что программы пользователей имеют различные объемы и особенности;
- повысить степень мультипрограммирования (в конечном счете - увеличить производительность ЭВМ).

Управление памятью

Механизмы управления памятью следующие:

- размещение с фиксированного адреса;
- размещение с любого адреса (перемещение программы в ОП в процессе исполнения);
- размещение программы вразброс (участками);
- частичное размещение программы в ОП.

Управление памятью

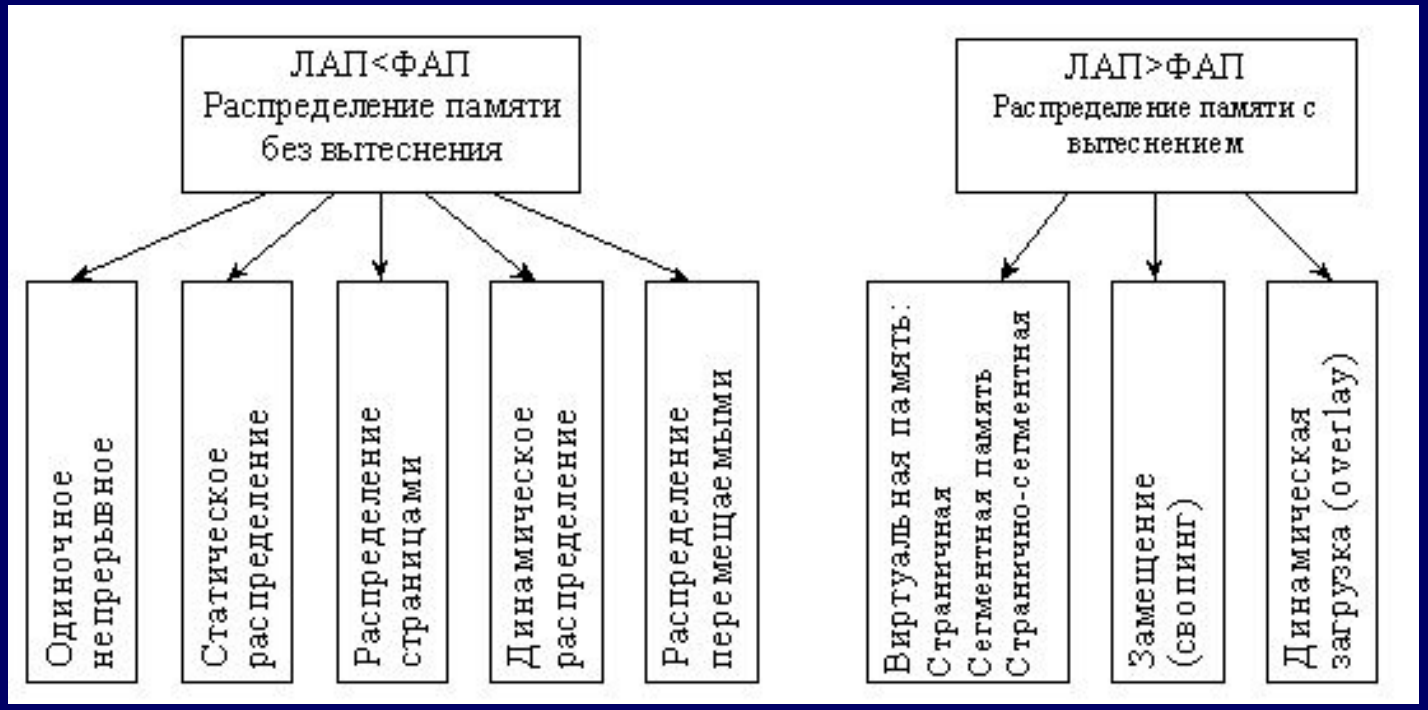


Схема управления оперативной памятью вычислительной системы

- ЛАП - логическое адресное пространство; ФАП - физическое адресное пространство.

Типы адресов

Для идентификации переменных и команд используются логические имена (метки), виртуальные адреса и физические адреса:

- Логические имена присваивает пользователь при написании программы на алгоритмическом языке или ассемблере.
- Виртуальные адреса вырабатывает транслятор, переводящий программу на машинный язык.

Типы адресов

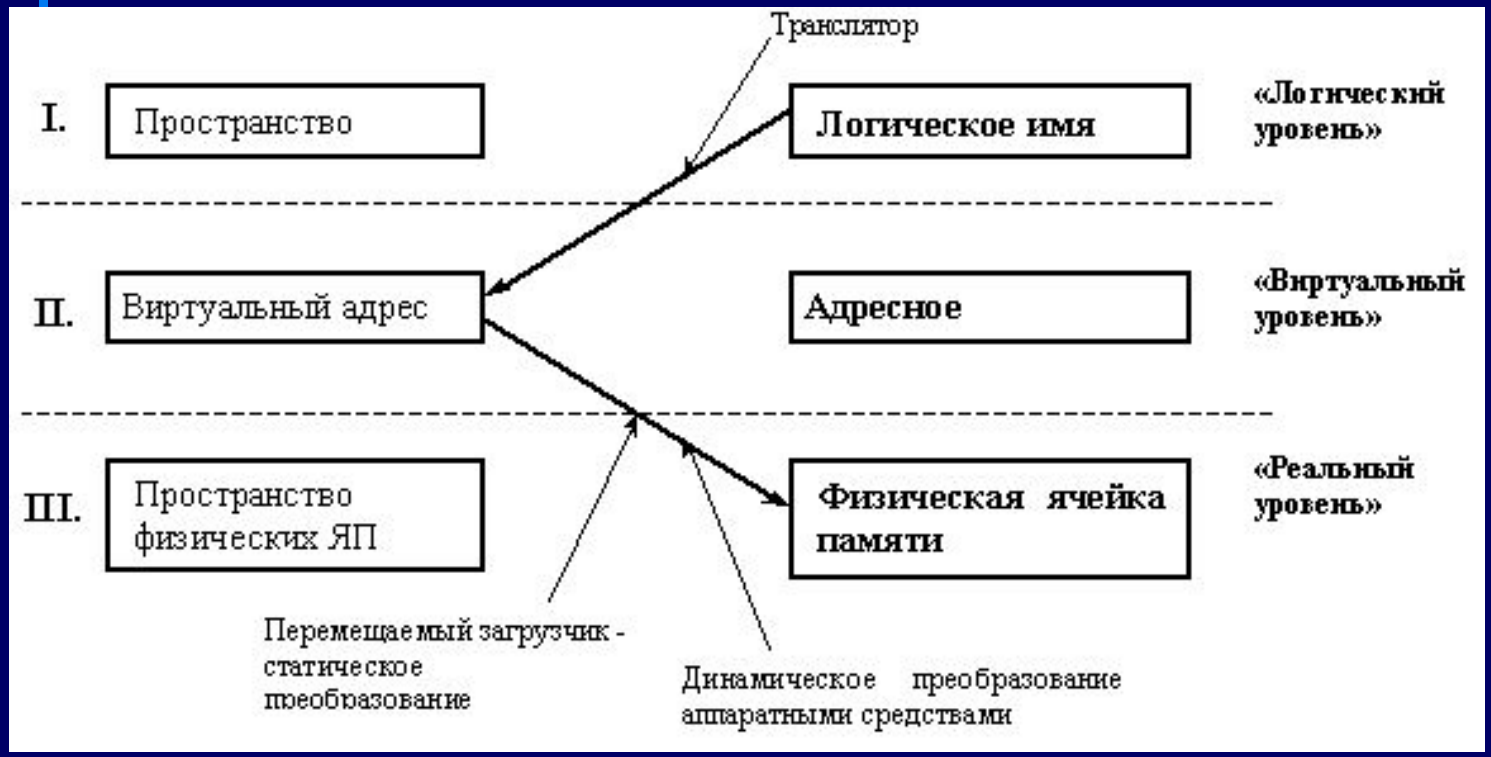


Схема "Пространство и отображение" при виртуальной памяти

- ЯП - ячейка памяти

Типы адресов

Отображение производится в два этапа:

- системой программирования;
- ОС с помощью программы управления памятью.

Виртуальная память

Уже достаточно давно пользователи столкнулись с проблемой размещения в памяти программ, размер которых превышал имеющуюся в наличии свободную память. Решением было разбиение программы на части, называемые оверлеями. 0-ой оверлей начинал выполняться первым. Когда он заканчивал свое выполнение, он вызывал другой оверлей. Все оверлеи хранились на диске и перемещались между памятью и диском средствами операционной системы.

Виртуальная память

ВИРТУАЛЬНЫМ

называется ресурс, который пользователю или пользовательской программе представляется обладающим свойствами, которыми он в действительности не обладает. Так, например, пользователю может быть предоставлена виртуальная оперативная память, размер которой превосходит всю имеющуюся в системе реальную оперативную память.

Виртуальная память

Таким образом, виртуальная память - это совокупность программно-аппаратных средств, позволяющих пользователям писать программы, размер которых превосходит имеющуюся оперативную память.

Виртуальная память

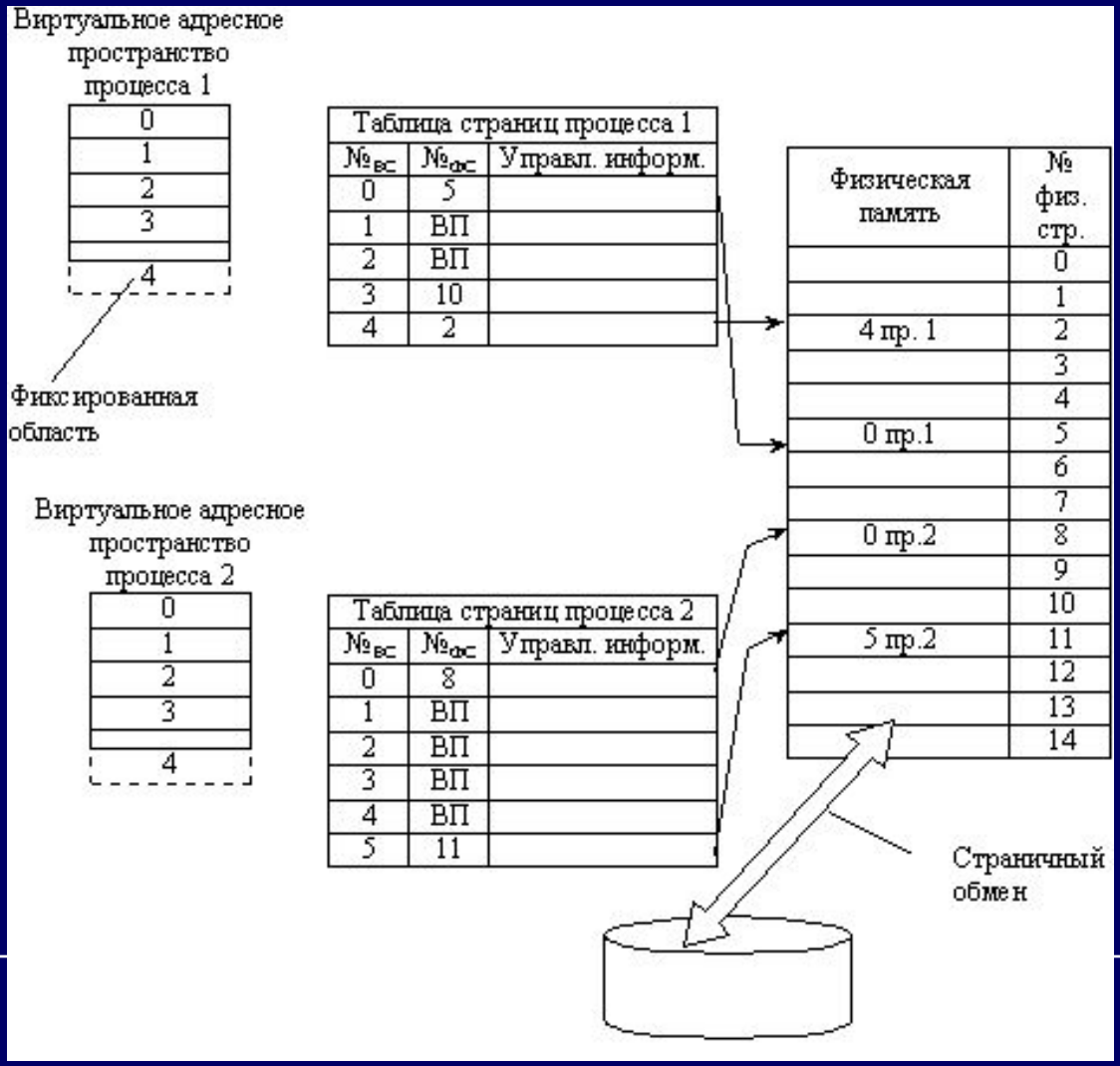
Для этого виртуальная память решает следующие задачи:

- размещает данные в запоминающих устройствах разного типа, например, часть программы в оперативной памяти, а часть на диске;
- перемещает по мере необходимости данные между запоминающими устройствами разного типа, например, подгружает нужную часть программы с диска в оперативную память;
- преобразует виртуальные адреса в физические.

Страничное распределение памяти

Виртуальное адресное пространство каждого процесса делится на части одинакового, фиксированного для данной системы размера, называемые виртуальными страницами. В общем случае размер виртуального адресного пространства не является кратным размеру страницы, поэтому последняя страница каждого процесса дополняется фиктивной областью.

Страничное распределение памяти



Страничное распределение памяти

При активизации очередного процесса в специальный регистр процессора загружается адрес таблицы страниц данного процесса. При каждом обращении к памяти происходит чтение из таблицы страниц информации о виртуальной странице, к которой произошло обращение. Если данная виртуальная страница находится в оперативной памяти, то выполняется преобразование виртуального адреса в физический. Если же нужная виртуальная страница в данный момент выгружена на диск, то происходит так называемое страничное прерывание. Выполняющийся процесс переводится в состояние ожидания, и активизируется другой процесс из очереди готовых. Параллельно программа обработки страничного прерывания находит на диске требуемую виртуальную страницу и пытается загрузить ее в оперативную память. Если в памяти имеется свободная физическая страница, то загрузка выполняется немедленно, если же свободных страниц нет, то решается вопрос, какую страницу следует выгрузить из оперативной памяти.

Страничное распределение памяти

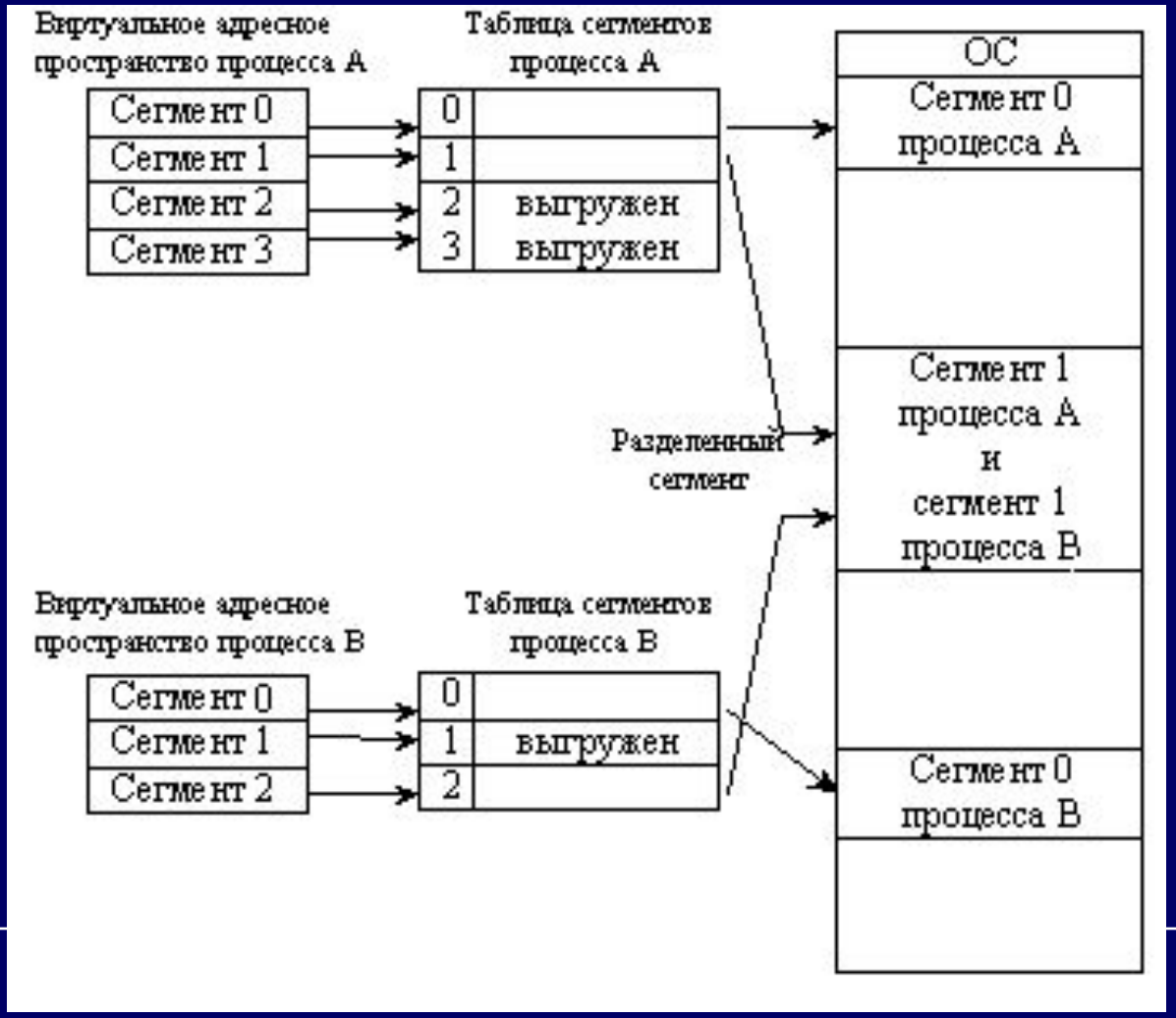
В данной ситуации может быть использовано много разных критериев выбора, наиболее популярные из них следующие:

- дольше всего не использовавшаяся страница,
- первая попавшаяся страница,
- страница, к которой в последнее время было меньше всего обращений.

Сегментное распределение

При страничной организации виртуальное адресное пространство процесса делится механически на равные части. Это не позволяет дифференцировать способы доступа к разным частям программы (сегментам), а это свойство часто бывает очень полезным. Например, можно запретить обращаться с операциями записи и чтения в кодовый сегмент программы, а для сегмента данных разрешить только чтение. Кроме того, разбиение программы на "осмысленные" части делает принципиально возможным разделение одного сегмента несколькими процессами.

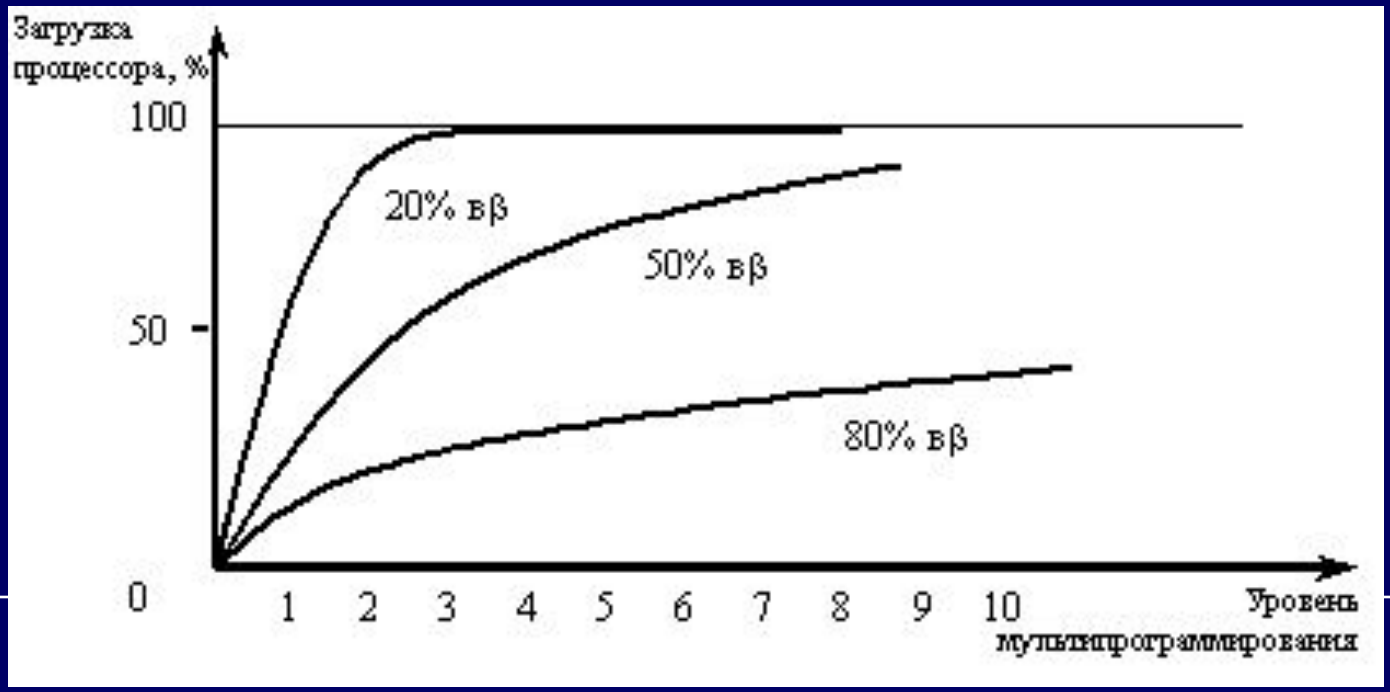
Распределение памяти сегментами



Сегментно-страничная организация памяти

Свопинг

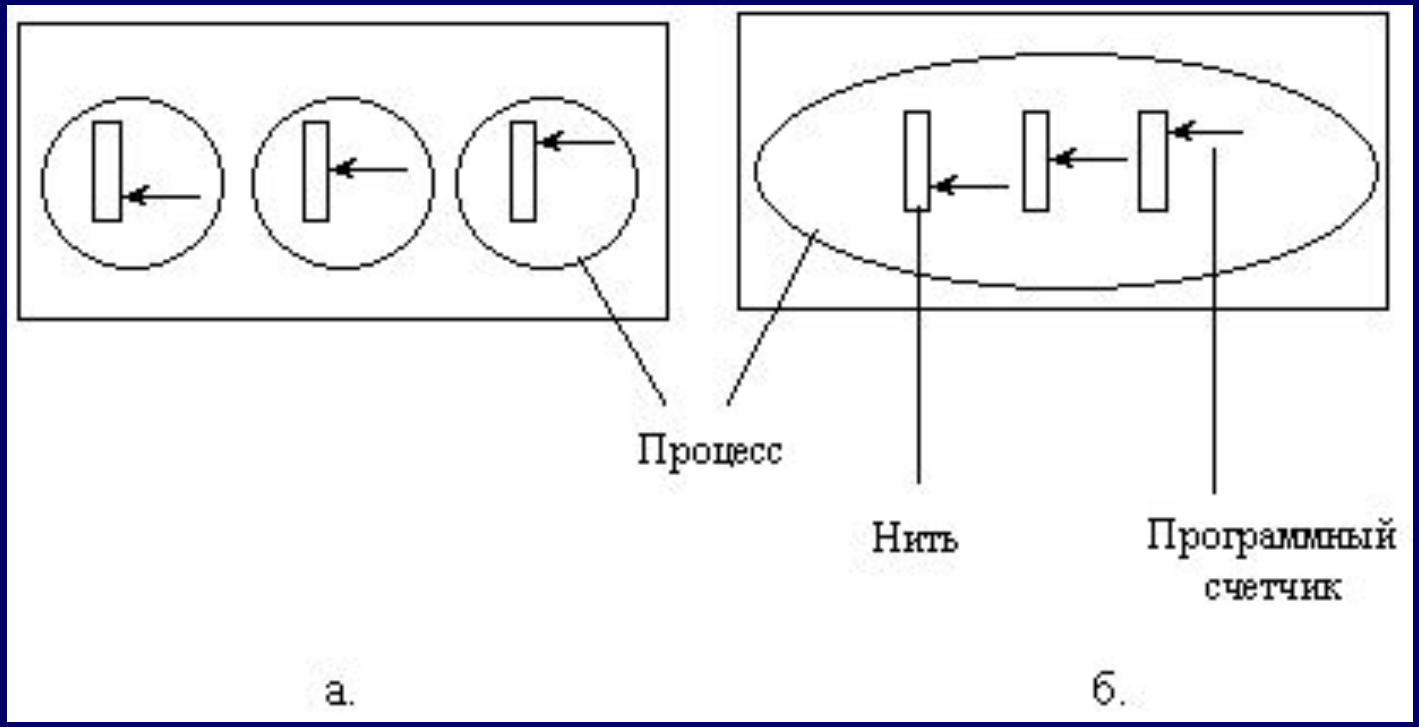
Зависимость загрузки процессора от числа задач и интенсивности ввода-вывода



Процессы и нити в распределенных системах

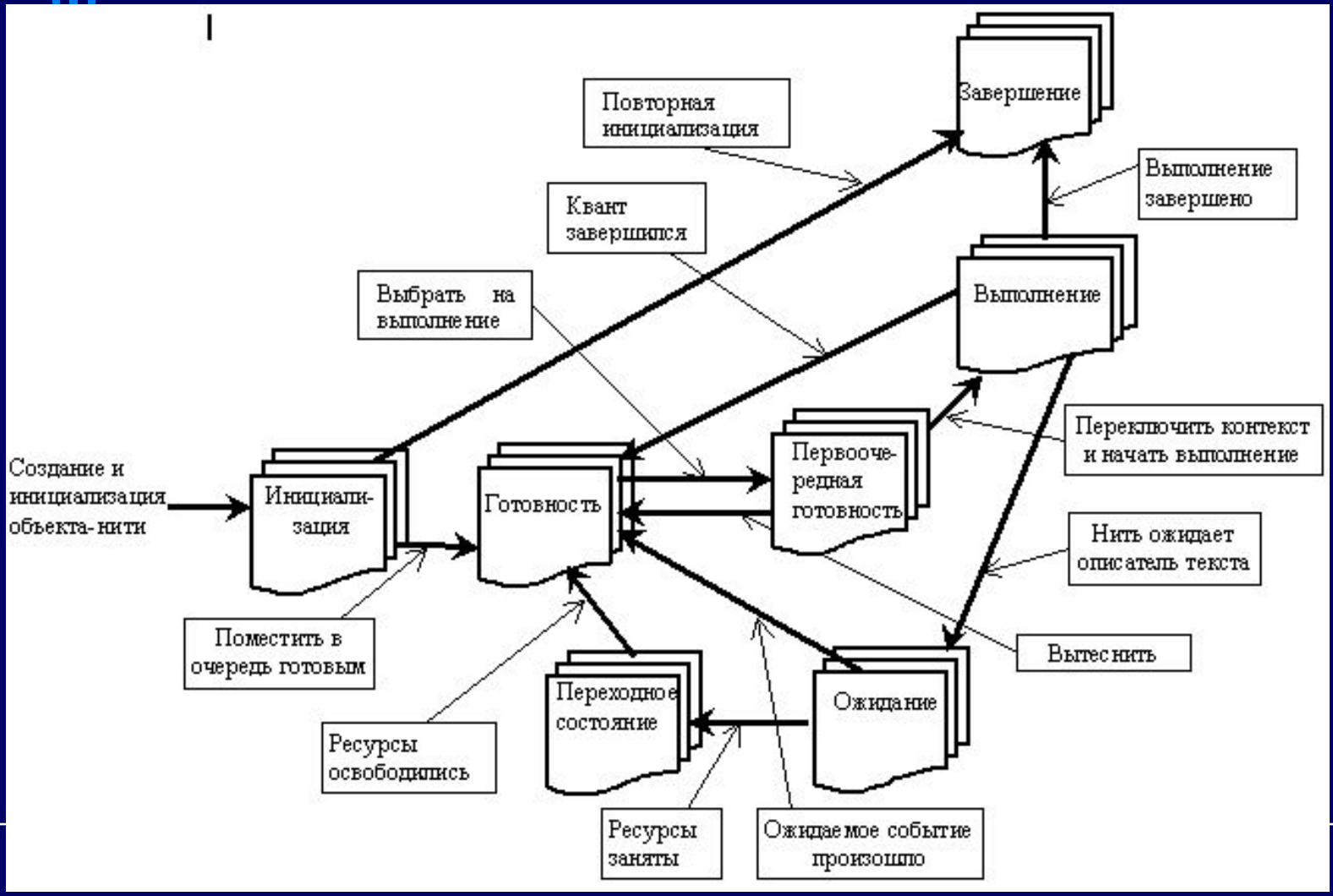
- В традиционных ОС понятие нити тождественно понятию процесса. В действительности желательно иметь несколько нитей управления, разделяющих единое адресное пространство, но выполняющихся квазипараллельно.
- Предположим, например, что файл-сервер блокируется, ожидания выполнения операции с диском. Если сервер имеет несколько нитей управления, вторая нить может выполняться, пока первая нить находится в состоянии ожидания. Это повышает пропускную способность и производительность. Эта цель не достигается путем создания двух независимых серверных процессов, потому что они должны разделять общий буфер кэша, который требуется им, чтобы быть в одном адресном пространстве.

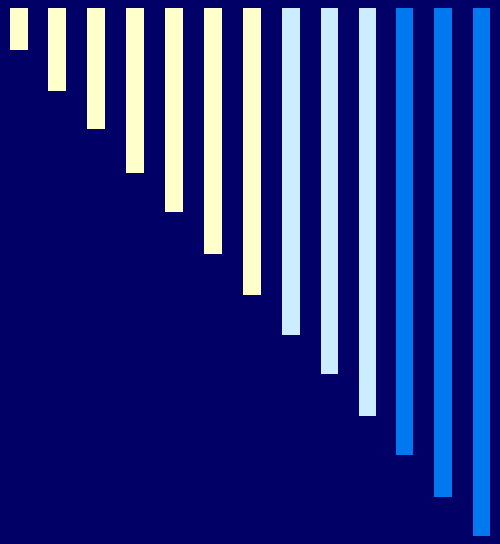
Сегментно-страничная организация памяти



а) три процесса с одной нитью каждый; б) один процесс с тремя нитями

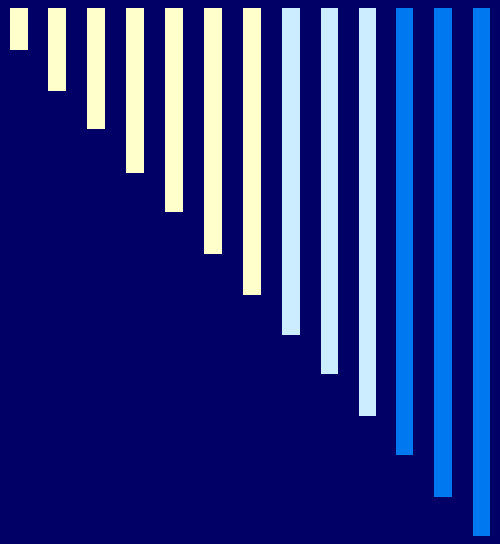
Граф состояний нити





НО КАК БЫ ЭТО НЕ БЫЛО СЛОЖНО – ПОМНИТЕ

Неразрешимыми являются только те задачи, за которые Вы не брались по-настоящему!!!



Спасибо за внимание!

Отзывы о лекции и информационная поддержка по адресу:

eart@ukr.net