ОПЕРАЦИОННАЯ ПЛАТФОРМА.

ОПРЕДЕЛЕНИЯ И КЛАССИФИКАЦИЯ

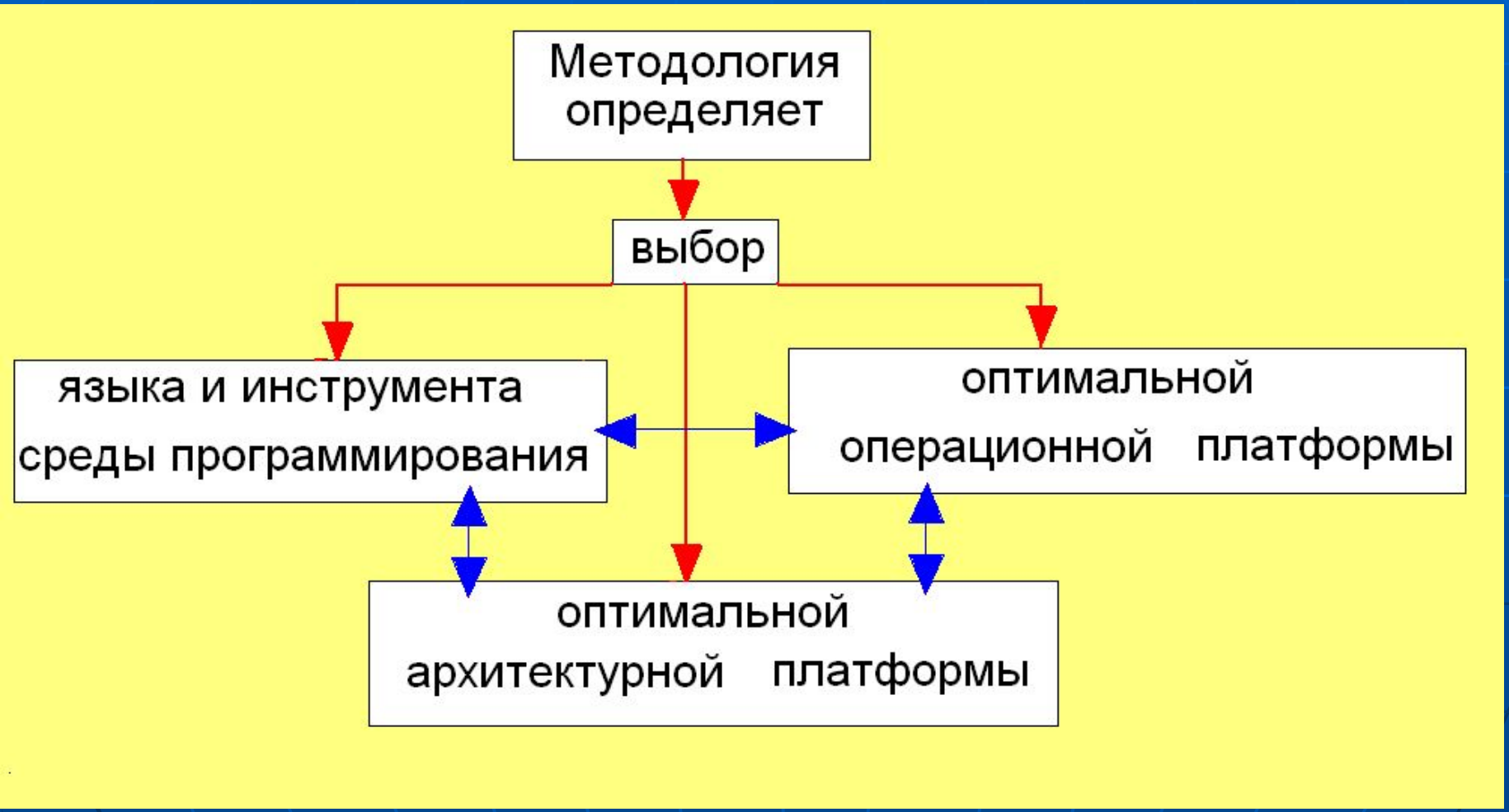
ЛЕКЦИЯ №3

Калинина А.П.

Содержание

1. Введение
2. Введение в операционные системы
3. Процессы и потоки (нити) управления

Операционная платформа – составляющая «системного» подхода



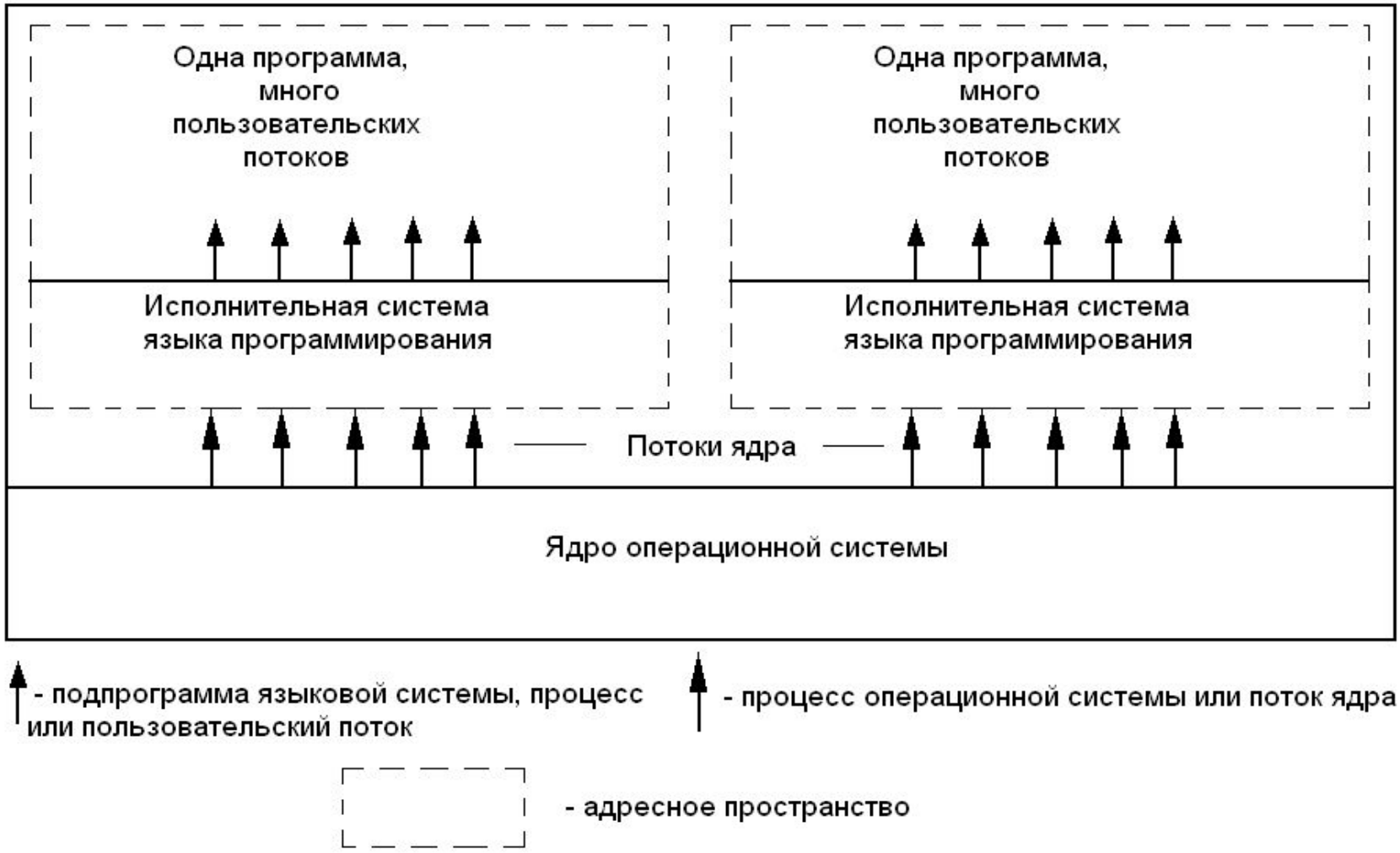
Необходимость изучения операционной платформы для любого программиста

- Основные идеи, концепции и алгоритмы, лежащие в основе операционных систем, применимы ко многим другим областям программирования, и особенно, к системному программированию;
- Операционная система – большая и сложная программа, на примере которой можно изучить вопросы создания сложных программных продуктов;
- Популярные программные продукты могут рассматриваться как надстройки над операционными системами (базы данных,…)

Особая необходимость изучения операционной платформы - для «параллельного» программиста

- современные операционные системы – по своей сути, параллельные программы
- технологии параллельного программирования – либо непосредственное использование сервисов операционной системы через интерфейс системных вызовов, либо надстройка над операционной системой

Пример: параллельный язык программирования с поддержкой пользовательских потоков как потоков ядра



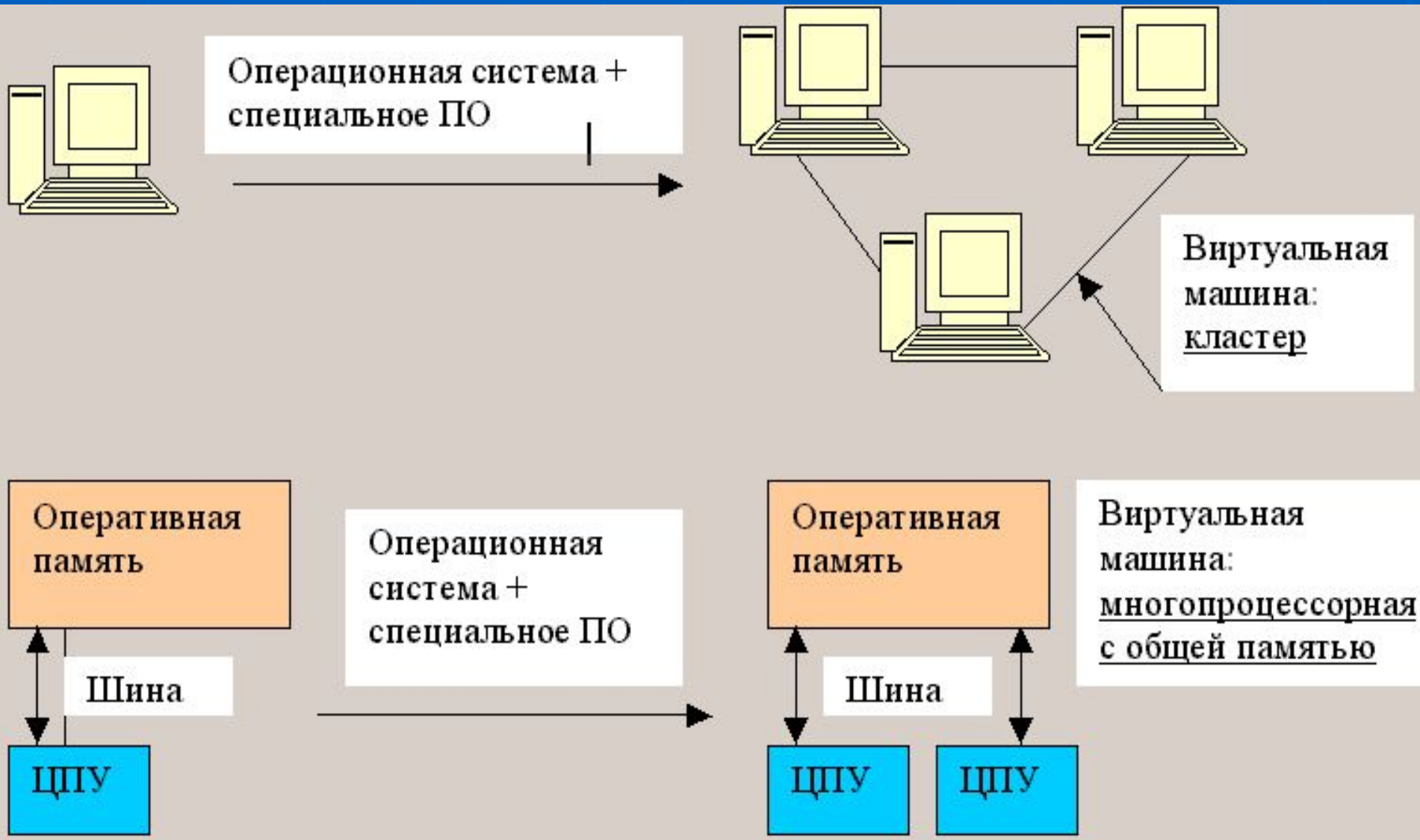
2. Введение в операционные системы

2.1. Основные понятия и определения
2.2. Поколения операционных систем
2.3. Классификация операционных систем

Основные функции операционной системы

- Образно говоря, основной функцией операционной системы можно считать чародейство – превращение системы в нечто большее, чем есть на самом деле.
- операционная система может создать иллюзию одновременного исполнения нескольких программ на одном процессоре.
- В итоге пользователь воспринимает виртуальную машину как компьютер, имеющий архитектуру, отличную от реально существующей.
- Программы с OpenMP и MPI можно тестировать на правильность работы на однопроцессорной машине, если есть подходящая операционная система и ПО.

Виртуальные параллельные архитектуры



Основные функции операционной системы

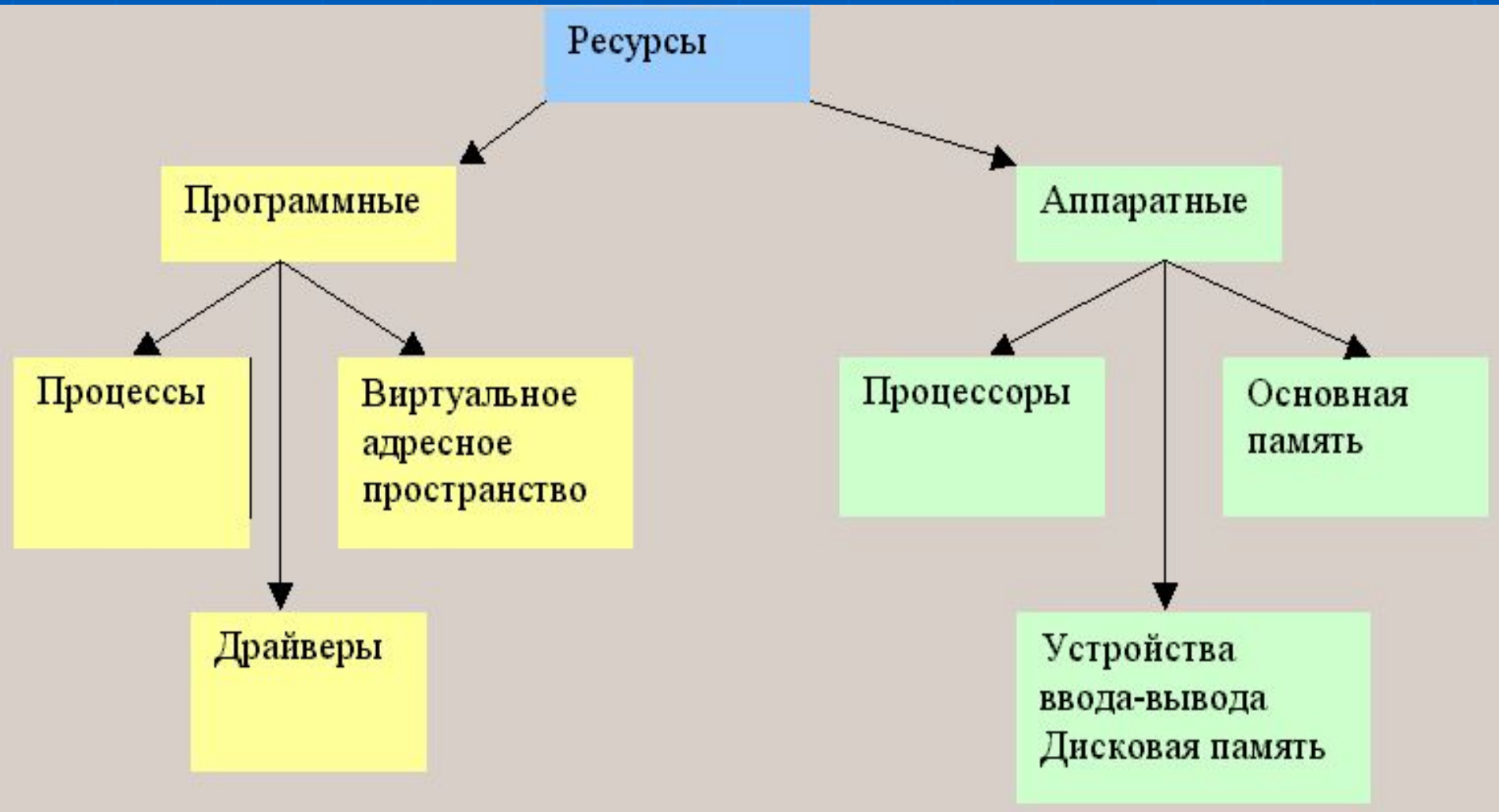
1. Предоставление пользователю-программисту вместо реальной аппаратуры компьютера расширенной виртуальной машины, с которой удобно работать.

2. Повышение эффективности использования компьютера за счет рационального управления его ресурсами.

О виртуальной машине

- ❑ Виртуальная машина – это вычислительная система заданной конфигурации, моделируемая для пользователя программными и аппаратными средствами конкретного реально существующего компьютера.
- ❑ Операционная система является тем слоем программного обеспечения, который преобразует аппаратную машину в виртуальную.
- ❑ Конфигурация виртуальной машины может существенно отличаться от реальной.

Классификация ресурсов



Составляющие операционной системы

- *Ядро операционной системы* – модули, выполняющие основные функции операционной системы.
- Эти модули обычно поддерживают управление процессами, памятью, устройствами ввода-вывода.
- Код ядра операционной системы исполняется в привилегированном режиме работы процессора.
- Обычные приложения в стандартном для данной операционной системе формате. Их называют *вспомогательными модулями* операционной системы.

2.2. Поколения операционных систем

- *Нулевое поколение*. В первых компьютерах операционные системы отсутствовали. Это период с момента появления первых компьютеров до середины 50- ых годов 20 века.
- *Первое поколение*. Пакетная обработка, мультипрограммные операционные системы. Появились в середине 50-ых годов 20 века.
- *Второе поколение*. Многорежимные операционные системы, операционные системы реального времени. Появились в середине 60-х гг. 20 века.
- *Третье поколение*. Операционные системы для персональных компьютеров, сетевые операционные системы. Появились в начале 80-х гг. 20 века.
- *Четвертое поколение*. Распределенные операционные системы. Появились в начале 90-х г 20 века.

2.3. Классификация операционных систем

2.3.1. Классификация по типу централизации

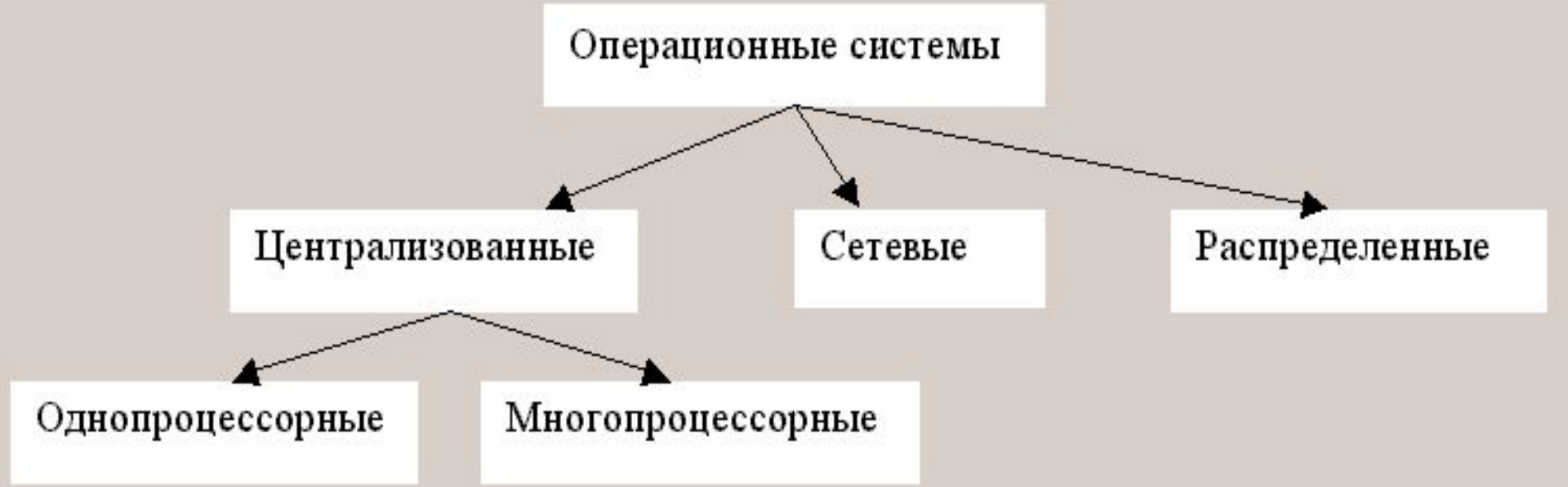
2.3.2. Классификация по особенностям алгоритмов управления ресурсами

2.3.4. Классификация по особенностям аппаратных платформ

2.3.5. Классификация по особенностям областей использования

2.3.6. Классификация по типу архитектуры ядра системы

2.3.1. Классификация по типу централизации



Распределенные операционные системы

- Предоставляют пользователю сети единую централизованную виртуальную машину, которая дает максимальную степень прозрачности сетевых ресурсов.

- Распределенные системы объединяют все компьютеров сети, для работы в тесной кооперации.

- При работе в таких системах пользователь, запускающий приложение, не знает, на каком компьютере оно реально выполняется.

Основная характеристика классификации параллельных и распределенных архитектур

наличие
общей или распределенной
(локальной для каждого из узлов)
памяти

Классификация вычислительных систем и ПО

Вычислительные системы можно разделить на два класса.

- *Системы с сильными связями*. Сюда относятся системы, состоящие из нескольких однородных процесооров и массива общей памяти.
- *Системы со слабыми связями*. Это системы, состоящие из однородных вычислительных узлов, каждый из которых имеет свою память.

Программное обеспечение можно также разделить на два класса.

- *Программное обеспечение с сильными связями*. Сюда относятся программы, которые при исполнении на нескольких вычислительных модулях в большой степени являются связанными между собой.
- *Программное обеспечение со слабыми связями*. Оно позволяет вычислительным модулям быть независимым друг от друга, но при необходимости взаимодействовать ограниченным количеством способов.

Обоснование классификации по типу централизации

Разновидности OpenMP

OpenMP для многопроцессорной системы с общей памятью

OpenMP для кластера

Малые отличия в синтаксисе

2.3.2. Классификация по особенностям алгоритмов управления ресурсами

- Поддержка многозадачности
- Поддержка многопользовательского режима
- Поддержка многопоточности
- Поддержка многопроцессорной обработки
- Концепция системы виртуальных машин (нескольких полноценных операционных систем)

2.3.4. Классификация по особенностям аппаратных платформ

- Операционные системы для мощных серверов
- Операционные системы для мощных станций и персональных компьютеров
- Операционные системы для карманных компьютеров

2.3.5. Классификация по особенностям областей использования

- Операционные системы пакетной обработки
- Операционные системы разделения времени. Такие системы предоставляют каждой из задач квант процессорного времени
- Операционные системы реального времени.

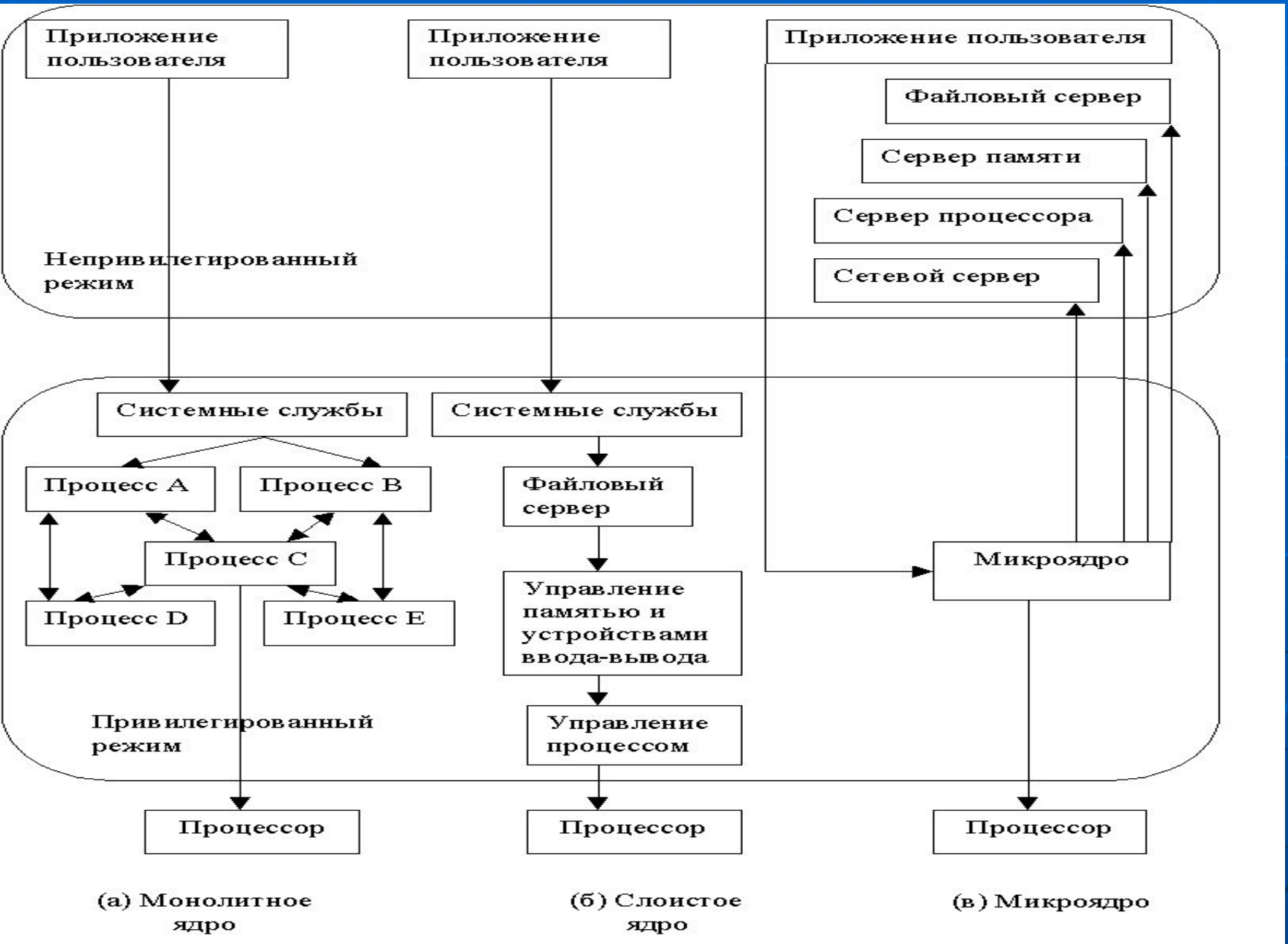
2.3.6. Классификация по типу архитектуры ядра системы

- Монолитное ядро. Такое ядро компонуется как одно программа, работающая в привилегированном режиме и использующая быстрые переходы с одной процедуры на другую

- Слоистое ядро. В этом случае компоненты операционной системы образуют уровни с хорошо продуманной функциональностью и интерфейсом. Как и в предыдущем случае, компоненты работают в привилегированном режиме

- Микроядро. Микроядро выполняет минимум функций по управлению аппаратурой. Обычно в него включаются машинно зависимые программы, некоторые функции управления процессами и

Классификация по типу архитектуры ядра



3. Процессы и потоки (нити) управления

- Процесс – это абстракция, описывающая выполняющуюся программу
- Процесс – это исполнение последовательности действий в среде, включающей собственно выполняющуюся программу, а также связанные с ней данные и состояния (открытые файлы, текущий каталог и т.п.)
- С точки зрения операционной системы процесс – это единица работы, заявка на потребление системных ресурсов
- Процесс – объект, которому выделяется процессор
- Процесс – это живая душа программы
- Первое упоминание о процессе появилось в 60-е годы 20 века в операционной системе MULTICS.

Основные состояния процесса

- Готовность – процесс находится в очереди на выполнение;
- Ожидание – процесс ожидает завершения события (например, освобождения ресурса);
- Остановлен – процесс остановлен, как правило, в отладочном режиме;
- Создание – выполнение действий, необходимых для создания процесса
- Завершение - выполнение действий, связанных с успешным завершением процесса
- Зомби – процесс закончен, но предок не принял его завершения.

Основные состояния процесса

Процессы с поддержкой многопоточности

- Поток (нить) управления– это исполнение команд программы в естественном порядке.
- Процессы делятся на традиционные - имеющие один поток управления и многопоточные (многонитевые).
- Многопоточность в рамках одного процесса имеет существенные преимущества. Переключение контекста между двумя потомками в одном процессе значительно проще, чем переключение контекста между двумя процессами.
- Все данные, за исключением стэка исполнения и содержимого регистра процессора, разделяются между потоками.

Классы программ, где необходима многопоточность

- Операционные системы
- Сетевые серверы
- Встроенные системы
- Вычислительные программы

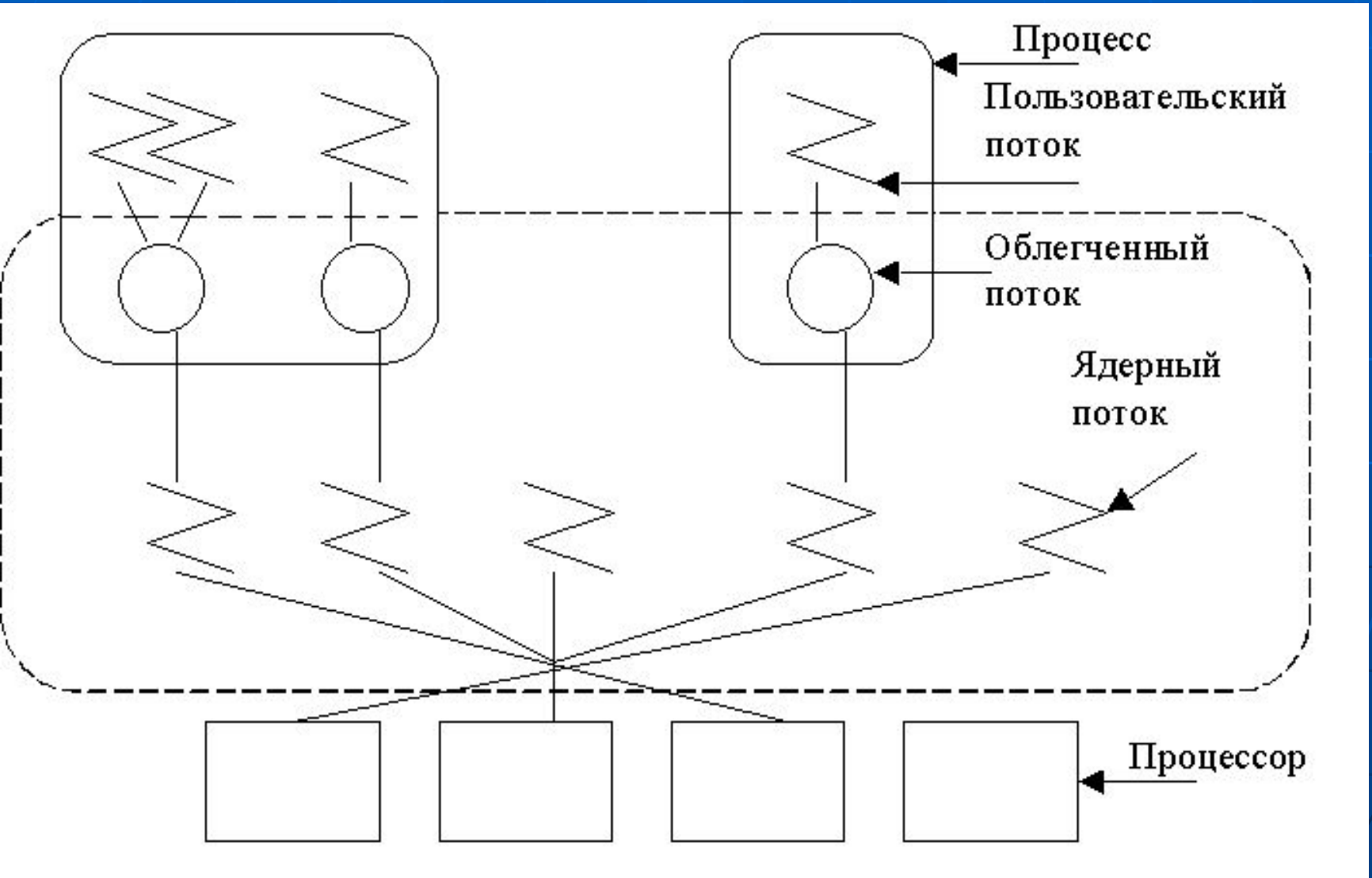
три класса потоков

- Потоки ядра являются базовыми потоками. Они располагаются в адресном пространстве ядра и непосредственно связаны с процессами.

- Облегченные (легковесные) потоки служат для организации пользователем нескольких потоков управления в адресном пространстве. Каждому облегченному потоку соответствует свой отдельный ядерный поток. Каждый облегченный поток может отдельно планироваться (на каждый процессор по потоку).

- Пользовательские потоки, для создания которых пользователь работает со стандартной библиотекой. Пользовательские потоки связываются с облегченными потоками.

Классы потоков



4. Windows – 2000 & Windows NT

- ❑ *процесс* представляет выполняющийся экземпляр программы. Он имеет собственное адресное пространство, где содержаться его код и данные.

- ❑ *Процесс* должен содержать минимум как один *поток*, так как именно он, а не процесс, является единицей планирования (данная операционная система относится к *системам разделения времени*, т.е. каждой единице предоставляется *квант* процессорного времени).

- ❑ *Процесс* может создавать несколько потоков, выполняемых в его адресном пространстве.

4. Windows – 2000 & Windows NT

- ❑ •…поддерживает особую легковесную форму потока, называемого *нитью*. В отличие от потоков, управление нитями осуществляется исключительно в пользовательском режиме, поэтому они не видны ни ядру, ни исполнительной системе.

- ❑ Приложение на основе нитей может эффективно использовать возможности мультипроцессора, если в нем задействовано более одного потока и реализован механизм управления параллельным выполнением нитей разных потоков

В Windows:

- Каждый пользовательский поток (не нить) регистрируется как поток ядра
- Создание параллельной секции – достаточно дорогостоящая операция
- Переключение контекста – дорогостоящая операция
- Разный тип потоков делает более выгодными разные параллельные конструкции

Задание 1.

1. Определить временные затраты, необходимые для создания параллельной секции. Для этого получить результат работы проекта Section_paral.

Время для создания параллельной секции определяется из сравнения времени работы следующих участков кода:

```
for (i=0; i<N; i++);
```

```
for (i=0; i<N; i++)
#pragma omp parallel
{
}
```

И составляет более 1000 циклов процессора

Задание 2.

1. Определить времена выполнения (и их разность) параллельных конструкций по проекту Critical_mutex

- #pragma omp critical
- omp_set_lock(&lsk);

Тест для определения временных затрат на создание критической секции

```
start = clock();

#pragma omp  parallel
    {
        for (i=0;i<N_critical; i++ )
        {

#pragma omp critical
            {
            }
        }
    }
    finish = clock();
```

Тест для определения временных затрат на создание omp_set_lock(&lsk);

```
omp_lock_t lsk;

   start = clock();
   omp_init_lock(&lsk);

#pragma omp  parallel   shared(lsk)
   {
      for (i=0;i<N_critical; i++ )
      {
         omp_set_lock(&lsk);
         {
         }
         omp_unset_lock(&lsk);
      }
   }
```