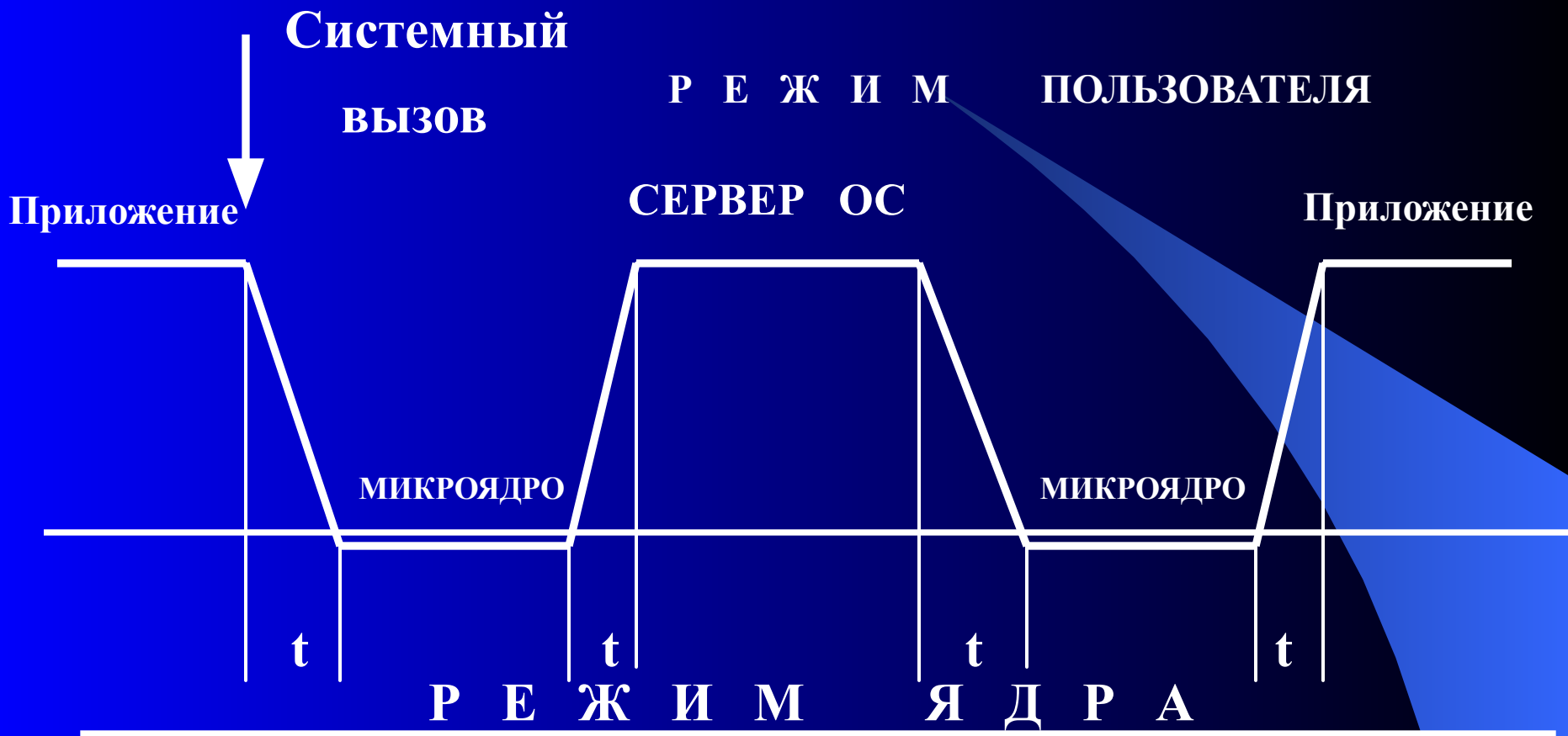
Структура ОС клиент-сервер

Смена режимов при выполнении вызова функции микроядра



Достоинства: единообразные интерфейсы, расширяемость, гибкость, переносимость, надежность, поддержка распределенных систем, поддержка объектно-ориентированных ОС.

Классификация ядер операционных систем

1. Наноядро (НЯ) – крайне упрощённое и минимальное ядро, выполняет лишь одну задачу – обработку аппаратных прерываний, генерируемых устройствами компьютера. После обработки посылает информацию о результатах обработки вышележащему программному обеспечению. Концепция наноядра близка к концепции HAL. НЯ используются для виртуализации аппаратного обеспечения реальных компьютеров или для реализации механизма гипервизора.

2. **Микроядро (МЯ)** предоставляет только элементарные функции управления процессами и минимальный набор абстракций для работы с оборудованием. Бóльшая часть работы осуществляется с помощью специальных пользовательских процессов, называемых сервисами. В микроядерной операционной системе можно, не прерывая ее работы, загружать и выгружать новые драйверы, файловые системы и т. д. Микроядерными являются ядра ОС Minix и GNU Hurd и ядро систем семейства BSD.

3. Экзоядро (ЭЯ) – предоставляет лишь набор сервисов для взаимодействия между приложениями, а также необходимый минимум функций, связанных с защитой: выделение и высвобождение ресурсов, контроль прав доступа, и т. д. ЭЯ не занимается предоставлением абстракций для физических ресурсов – эти функции выносятся в библиотеку пользовательского уровня (так называемую libOS). В отличие от микроядра ОС, базирующиеся на ЭЯ, обеспечивают большую эффективность за счет отсутствия необходимости в переключении между процессами при каждом обращении к оборудованию.

4. Монолитное ядро (МЯ) предоставляет широкий набор абстракций оборудования. Все части ядра работают в одном адресном пространстве. МЯ требуют перекомпиляции при изменении состава оборудования. Компоненты операционной системы являются не самостоятельными модулями, а составными частями одной программы. МЯ более производительно, чем микроядро, поскольку работает как один большой процесс. МЯ является большинство Unix-систем и Linux. Монолитность ядер усложняет отладку, понимание кода ядра, добавление новых функций и возможностей, удаление ненужного, унаследованного от предыдущих версий, кода. «Разбухание» кода монолитных ядер также повышает требования к объёму оперативной памяти.

5. Модульное ядро (Мод. Я) – современная, усовершенствованная модификация архитектуры МЯ. В отличие от «классических» МЯ, модульные ядра не требуют полной перекомпиляции ядра при изменении состава аппаратного обеспечения компьютера. Вместо этого они предоставляют тот или иной механизм подгрузки модулей, поддерживающих то или иное аппаратное обеспечение (например, драйверов). Подгрузка модулей может быть как динамической, так и статической (при перезагрузке ОС после переконфигурирования системы). Мод. Я удобнее для разработки, чем традиционные монолитные ядра. Они пре-доставляют программный интерфейс (API) для связывания модулей с ядром, для обеспечения динамической подгрузки и выгрузки модулей. Не все части ядра могут быть сделаны модулями. Некоторые части ядра всегда обязаны присутствовать в оперативной памяти и должны быть жёстко «вшиты» в ядро.

6. Гибридное ядро (ГЯ) – модифицированные микроядра, позволяющие для ускорения работы запускать «несущественные» части в пространстве яд-ра. Имеют «гибридные» достоинства и недостатки. примером смешанного подхода может служить возможность запуска операционной системы с монолитным ядром под управлением микроядра. Так устроены 4.4BSD и MkLinux, основанные на микроядре Mach. Микроядро обеспечивает управление виртуальной памятью и работу низкоуровневых драйверов. Все остальные функции, в том числе взаимодействие с прикладными программами, осуществляется моно-литным ядром. Данный подход сформировался в результате попыток исполь-зовать преимущества микроядерной архитектуры, сохраняя по возможности хорошо отлаженный код монолитного ядра.

Наиболее тесно элементы микроядерной архитектуры и элементы монолитного ядра переплетены в ядре Windows NT. Хотя Windows NT часто называют микроядерной операционной системой, это не совсем так. Микроядро NT слишком велико (более 1 Мбайт), чтобы носить приставку «микро». Компоненты ядра Windows NT располагаются в вытесняемой памяти и взаимодействуют друг с другом путем передачи сообщений, как и положено в микроядерных операционных системах. В то же время все компоненты ядра работают в одном адресном пространстве и активно используют общие структуры данных, что свойственно операционным системам с монолитным ядром.

Средства аппаратной поддержки ОС

1. Средства поддержки привилегированного режима: системные регистры процессора, слово состояния процессора, привилегированные команды, привилегированные режимы.

2. Средства трансляции адресов: буферы быстрой трансляции виртуальных адресов, регистры процессора, средства поддержки сегментно-страничных таблиц.

3. Средства переключения процессов: регистры общего назначения, системные регистры и указатели, флаги операций.

4. Система прерываний: регистры и флаги прерываний, регистры масок, контроллеры прерываний.

5. Системный таймер и системные часы.

6. Средства защиты памяти: граничные регистры, ключи.

1.5. Классификация операционных систем

1. **Назначение** (универсальные, специализированные – управление производством, обучение)
2. **Способ загрузки** (загружаемые, постоянно находящиеся в памяти)
3. **Особенности алгоритмов управления ресурсами**
   - 3.1. Многозадачность: однозадачные (MS DOS), невытесняющая многозадачность (Windows 3.x, NewWare), вытесняющая многозадачность (Windows NT, OS/2, Unix)
   - 3.2. Многопользовательский режим: отсутствие (MS DOS, Windows 3.x), имеется (Windows NT, OS/2, Unix)
   - 3.3. Многопроцессорная обработка: отсутствие, асимметричные ОС, симметричные ОС
4. **По базовой технологии** (Юникс-подобные или подобные Windows)
5. **По типу лицензии** (проприетарная или открытая)
6. **По состоянию развития** (устаревшая DOS, NextStep или современные GNU/Linux и Windows)

7. Область использования и форма эксплуатации

пакетная обработка (OS/360)

разделение времени

реальное время (VxWorks,QNX)

8. Аппаратная платформа

8.1. ОС для смарт-карт (с интерпретатором виртуальной Java-машины)

8.2. Встроенные ОС (Palm OS, Windows CE –Consumer Electronics)

8.3. ОС для ПК (Windows 9.x, Windows 2000, Linux, Mac OS X)

8.4. ОС мини-ЭВМ (RT-11 и RSX-11M для PDP-11, UNIX для PDP-7)

8.5. ОС мэйнфреймов (OS/390 – пакетная обработка, разделение времени, обработка транзакций)

8.6. Серверные операционные системы для ЛВС, Интранет и Интернет (UNIX, AIX, Windows 2000/2002, Linux)

8.7. Кластерные операционные системы (Windows 2000 Cluster Server, Sun Cluster (Solaris))

1.6. Эффективность и требования, предъявляемые к операционным системам

1. Эффективность – степень соответствия своему назначению, техническое совершенство и экономическая целесообразность
2. Надежность и отказоустойчивость
3. Безопасность (защищенность)
4. Предсказуемость
5. Расширяемость
6. Переносимость
7. Совместимость
8. Удобство
9. Масштабируемость

1.7. Множественные прикладные среды. Совместимость

Совместимость – возможность операционной системы выполнять приложения , разработанные для других операционных систем.

Виды совместимости:

1. На двоичном уровне (уровень исполняемой программы).
2. На уровне исходных текстов (уровень исходного модуля).

Вид совместимости определяется:

1. Архитектурой центрального процессора.
2. Интерфейсом прикладного программирования (API).
3. Внутренней структурой исполняемого файла.
4. Наличием соответствующих компиляторов и библиотек.

Способы достижения совместимости:

1. Эмуляция двоичного кода.
2. Трансляция библиотек.
3. Создание множественных прикладных сред различной архитектуры.