Операционные среды, системы и оболочки

Тема 7. Сетевые операционные системы

Автор : доктор технических наук, профессор Назаров С.В.

Тема 7. Сетевые операционные системы

7.1. Сетевые и распределенные операционные системы

7.2. Виды сетевых операционных систем

7.3. Требования, предъявляемые к сетевым операционным системам

7.3. Требования, предъявляемые к корпоративным сетевым операционным системам

7.4. Серверные операционные системы ведущих производителей

7.5. Тенденции на рынке операционных систем

7.6. Операционная система UNIX

7.7. Операционная система Windows 2000

Литература:

Базовый учебник стр. 330 – 383;

Л1 стр. 39 – 42;

Л2 стр. 115 – 135;

Л4 стр. 736 – 776, 836 – 869, 910 - 930

7.1. Сетевые и распределенные операционные системы

Сетевая ОС предоставляет пользователю виртуальную вычислительную систему, работать с которой проще, чем с реальной сетевой аппаратурой. В то же время эта виртуальная система не полностью скрывает распределенную природу своего реального прототипа.

Термин “сетевая операционная система” используется в двух значениях: 1. Совокупность взаимодействующих ОС всех компьютеров сети. 2. Операционная система отдельного компьютера, позволяющая ему работать в сети.

В идеальном случае сетевая ОС должна предоставлять пользователю сетевые ресурсы в виде ресурсов единой централизованной виртуальной машины. В этом случае сетевая ОС является распределенной ОС. Распределенная операционная система существует как единая ОС в масштабах всей вычислительной системы.

Степень автономности каждого компьютера сети, работающего под управлением сетевой ОС, значительно выше по сравнению с с компьютерами, работающими под управлением распределенной ОС.

7.2. Виды сетевых операционных систем

1. Сети отделов – используются небольшой группой сотрудников, решающих общие задачи. Имеют 1-2 файловых сервера и не более30 пользователей. Задачи сетевой ОС: разделение локальных ресурсов (приложений, данных, принтеров, модемов).

2. Сети кампусов – соединяют несколько сетей отделов внутри одной территории предприятия. Задачи сетевой ОС: взаимодействие между сетями отделов, доступ к базам данных предприятия, доступ к факс-серверам и серверам скоростных модемов, высокоскоростных принтеров и др.

3. Сети предприятия (корпоративные сети) – объединяют все компьютеры всех территорий отдельного предприятия. Задачи сетевой ОС: предоставлять доступ к информации и приложениям, находящимся в других рабочих группах, других отделах, подразделениях и штаб-квартирах корпорации, обеспечивать широкий набор сервисов –справочную и почтовую службы, средства коллективной работы,поддержку удаленных пользователей, факс-сервис, обработку голосовых сообщений, организацию видеоконференций и др. Особую важность приобретают вопросы безопасности по причинам, связанным с крупномасштабностью сети.

7.3. Требования, предъявляемые к корпоративным сетевым операционным системам

1. Масштабируемость, т.е. способность обеспечивать работу в широком диапазоне различных количественных характеристик сети.
2. Совместимость с другими продуктами, способность работать в сложной гетерогенной среде интерсети в режиме plug-and-play.
3. Поддержка многообразных ОС конечных пользователей (DOS, UNIX, OS/2, Mac, Windows).
4. Поддержка нескольких стеков протоколов (TCP/IP, IPX/SPX, NetBIOS, DECnet, AppleTalk, OSI), обеспечение простого доступа к удаленным ресурсам и удобных процедур управления сервисами.
5. Поддержка многосерверной сети и эффективная интеграция с другими операционными системами.
6. Наличие централизованной масштабируемой справочной службы.
7. Развитая система сервисов: файл-сервис, принт-сервис, безопасность данных и отказоустойчивость, архивирование данных,служба обмена сообщениями, разнообразные базы данных, вызов удаленных процедур RPC и др.
8. Поддержка сетевого оборудования различных стандартов (Ethernet, Token Ring, ARCnet, FDDI), поддержка стандартов управления сетью.

7.4. Серверные операционные системы ведущих производителей

Windows (Microsoft)

Windows NT. **Применение Windows NT Server 4.0 в качестве серверной операционной системы во многих случаях было экономически оправданным, что сделало данную операционную систему весьма популярной у малых и средних предприятий — она до сих пор активно используется многими компаниями.**

Windows 2000. **Windows 2000 является самой популярной операционной системой Microsoft в корпоративном секторе. К серверным операционным системам этого семейства относятся *Windows 2000 Server* — универсальная сетевая операционная система для серверов рабочих групп и отделов, *Windows 2000 Advanced Server* — операционная система для эксплуатации бизнес-приложений и приложений для электронной коммерции и *Windows 2000 Datacenter Server* — ОС для наиболее ответственных приложений обработки данных.**

Windows 2000 Advanced Server **поддерживает кластеризацию и баланс нагрузки, что делает возможным выполнение масштабируемых приложений с непрерывным доступом к данным. *Windows 2000 Datacenter Server* поддерживает симметричную мультипроцессорную обработку с использованием 32 процессоров, 64 Гбайт оперативной памяти, средства восстановления после отказа на основе четырехузловой кластеризации.**

Windows Server 2003

- ***Windows Server 2003 Web Edition*** **— операционная система для поддержки Web-приложений и Web-сервисов, включая приложения ASP .NET (Active Server Pages);**

- ***Windows Server 2003 Standard Edition*** **— сетевая операционная система для выполнения серверной части бизнес-решений и рассчитанная на применение в небольших компаниях и подразделениях, поддерживает до 4 Гбайт оперативной памяти и симметричную многопроцессорную обработки с использованием двух процессоров;**

- ***Windows Server 2003 Enterprise Edition*** **— предназначена для средних и крупных компаний. Она поддерживает серверы на базе 64-разрядных процессоров (до восьми штук) и объем оперативной памяти до 64 Гбайт и выпускается в версиях для 32- и 64-разрядных платформ;**

- ***Windows Server 2003 Datacenter Edition*** **— предназначена для создания критически важных технических решений с высокими требованиями к масштабируемости и доступности. К таким решениям относятся приложения для обработки транзакций в режиме реального времени, а также решения, основанные на интеграции нескольких серверных продуктов. В ОС реализована поддержка симметричной многопроцессорной обработки (до 32 процессоров), а также имеются службы балансировки нагрузки и создания кластеров, состоящих из восьми узлов. Эта ОС доступна для 32- и 64-разрядных платформ.**

UNIX

Solaris (Sun Microsystems). ***Sun Solaris*** **сегодня входит в число самых известных коммерческих версий UNIX. ОС обладает развитыми средствами поддержки сетевого взаимодействия и представляет собой одну из самых популярных платформ для разработки корпоративных решений — для нее существует около 12 тыс. различных приложений, в том числе серверов приложений и СУБД почти от всех ведущих производителей. ОС Solaris 9 поддерживает до 1 млн. работающих процессов, до 128 процессоров в одной системе и до 848 процессоров в кластере, до 576 Гбайт физической оперативной памяти, поддержку файловых систем размером до 252 Тбайт, наличие средств управления конфигурациями и изменениями, встроенную совместимость с Linux.**

HP-UX (Hewlett-Packard). ***HP-UX 11i*** **имеет средства интеграции с Windows и Linux, средства переноса Java-приложений, разработанных для этих платформ, а также средства повышения производительности Java-приложений. HP-UX 11i поддерживает Linux API, что гарантирует перенос приложений между HP-UX и Linux. Операционная система поддерживает до 256 процессоров и кластеры размером до 128 узлов, подключение и отключение дополнительных процессоров, замену аппаратного обеспечения, динамическую настройку и обновление операционной системы без необходимости перезагрузки, резервное копирование в режиме on-line и дефрагментацию дисков без выключения системы.**

AIX (IBM). AIX является клоном UNIX производства IBM, предназначенным для выполнения на серверах IBM @server pSeries и RS/6000. Операционная система обладает совместимостью с Linux. В числе особенностей ***AIX 5L*** — наличие полностью 64-разрядных ядра, драйверов устройств и среды исполнения приложений (при этом имеется и 32-разрядное ядро и поддержка 32-рязрядных приложений), поддержка 256 Гбайт оперативной памяти, поддержка файлов объемом до 1 Тбайт, удобные средства администрирования, поддержка кластеров (до 32 компьютеров), развитые средства сетевой поддержки.

Linux и FreeBSD. Операционная система ***Linux*** — это некоммерческий продукт категории Open Source для платформы Intel. Список серверных продуктов для Linux включает такие популярные продукты, как Web-сервер Apache, серверные СУБД и серверы приложений практически от всех производителей.

Еще одной известной некоммерческой версией UNIX является ***FreeBSD***, доступная для платформ Intel и DEC Alpha. Основой FreeBSD послужил дистрибутив BSD UNIX, выпущенный группой Калифорнийского университета (Беркли). Операционная система имеет объединенный кэш виртуальной памяти и буферов файловых систем, совместно используемые библиотеки, модули совместимости с приложениями других версий UNIX, динамически загружаемые модули ядра, позволяющие добавлять во время работы поддержку новых типов файловых систем, сетевых протоколов или эмуляторов без перегенерации ядра.

NetWare (Novell)

Основными особенностями последней версии операционной системы, *Novell NetWare 6.5*, являются возможность создания географически распределенных кластеров, наличие средств поддержки мобильных и удаленных пользователей, инструментов управления удаленными сетевыми ресурсами, а также средств синхронизации информации о пользователях и приведения в соответствие между собой каталогов в смешанных средах. Защита данных в *Novell NetWare 6.5* осуществляется с помощью служб каталогов NDS eDirectory. Данная операционная система обычно применяется в качестве сетевого и файлового сервера, сервера печати и групповой работы.

Mac OS X (Apple)

Операционная система *Mac OS X*, созданная компанией Apple совместно с рядом университетских ученых, основана на BSD UNIX. В 1999 году версия *Mac OS X Server* была выпущена в виде продукта Open Source, что позволило разработчикам адаптировать Mac OS X для конкретных заказчиков, а также привлечь их к дальнейшему развитию этой операционной системы. *Mac OS X* характеризуется наличием менеджера виртуальной памяти, возможностью полной изоляции приложений друг от друга, поддержкой многозадачности, сравнимой с аналогичной поддержкой в Windows.

Операционная система Z/OC для высокоуровневых вычислительных устройств eServer z900 (IBM)

Z/OS является усовершенствованной версией операционной системы OS/390, однако по сравнению с последней новая ОС имеет ряд преимуществ. Среди них zOS – поддержка 64-разрядной адресации, позволяющей ускорить обмен данными между модулями памяти и процессорами и увеличить производительность работы с большими базами данных.

Важным компонентом z/OS является ПО Intelligent Resource Director, предназначенное для автоматического распределения вычислительных мощностей между одновременно выполняющимися приложениями. Другими отличительными особенностями новой ОС является поддержка ПО для платформ Java и Linux, а также простота в настройке и администрировании.

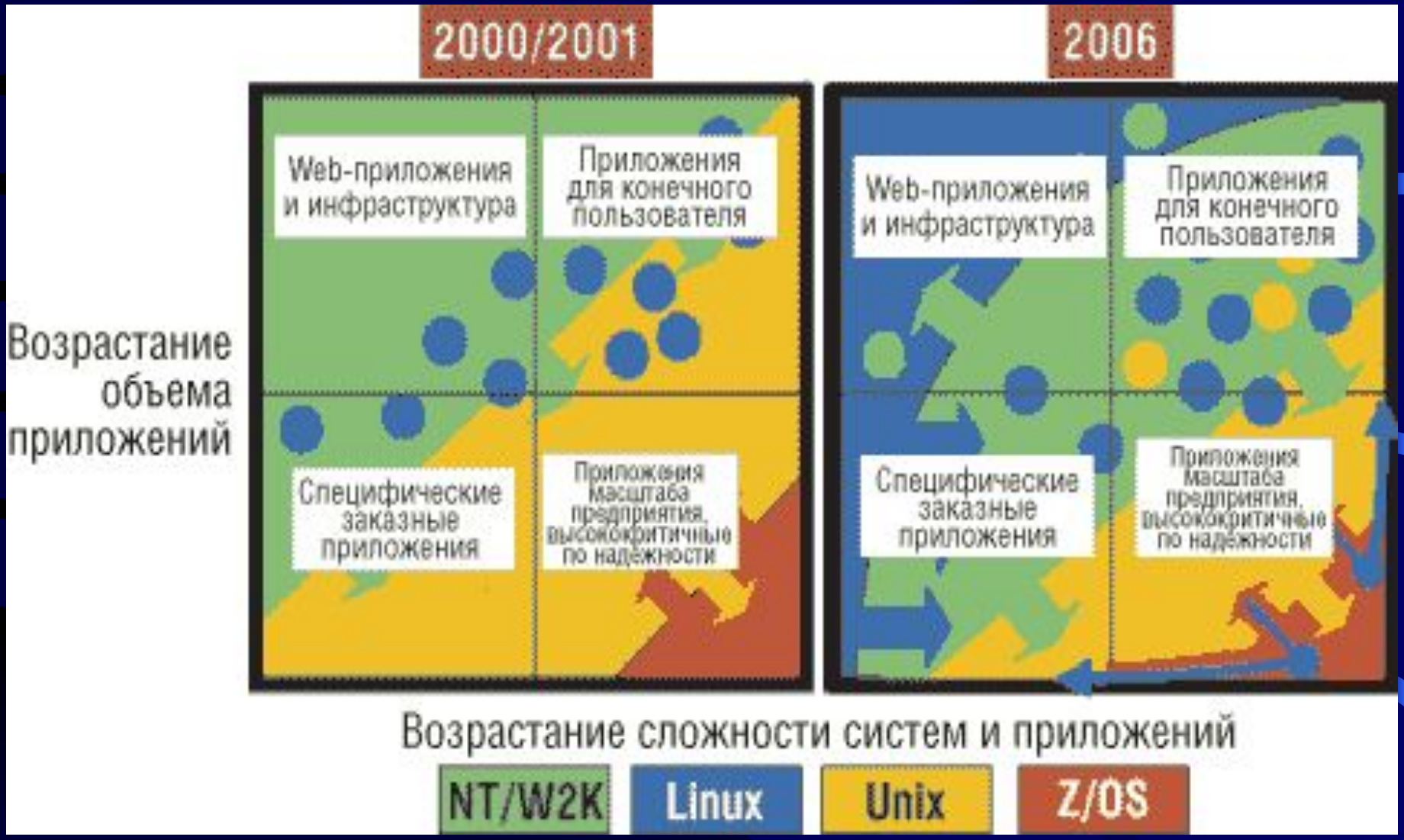
Операционная система IBM i (ранее известная как i5/OS)

работающая на сервере IBM Power Systems, предоставляет высокомасштабируемую и устойчивую к воздействию вирусов архитектуру, подтвердившую свою исключительную надежность для работы бизнес-приложений. ОС IBM i представляет собой надежное сочетание реляционной базы данных, функций обеспечения безопасности, веб-сервисов, средств организации сети и средств управления хранением данных.

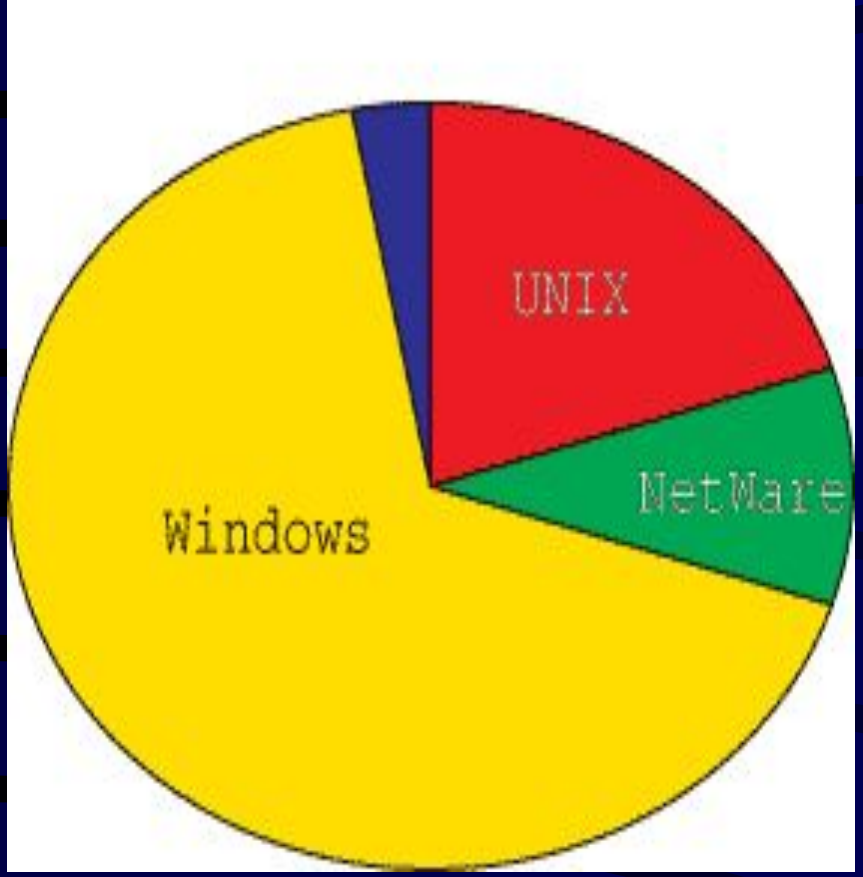
Внедрение инноваций:

- **Поддержка серверов POWER™ с числом процессоров от 1 до 64 и blade-серверов**
- **Оптимизированная интеграция с включенной в поставку базой данных DB2® (US)**
- **Ориентация на открытые приложения (среды выполнения приложений IBM Rational®, C, RPG, COBOL, C++, Java™, PHP, SOA, UNIX®) (US)**
- **Поддержка одновременного выполнения нескольких критически важных приложений**
- **Динамическое управление рабочей нагрузкой для распределения мощности между участками и периодами максимального потребления**
- **Поддержка EnergyScale™ для эффективного управления энергопотреблением**
- **Новый адаптер Fibre Channel с передовым уровнем производительности**
- **Виртуализация обработки и хранения для получения максимальной выгоды от инвестиций в оборудование и приобретенных лицензий**

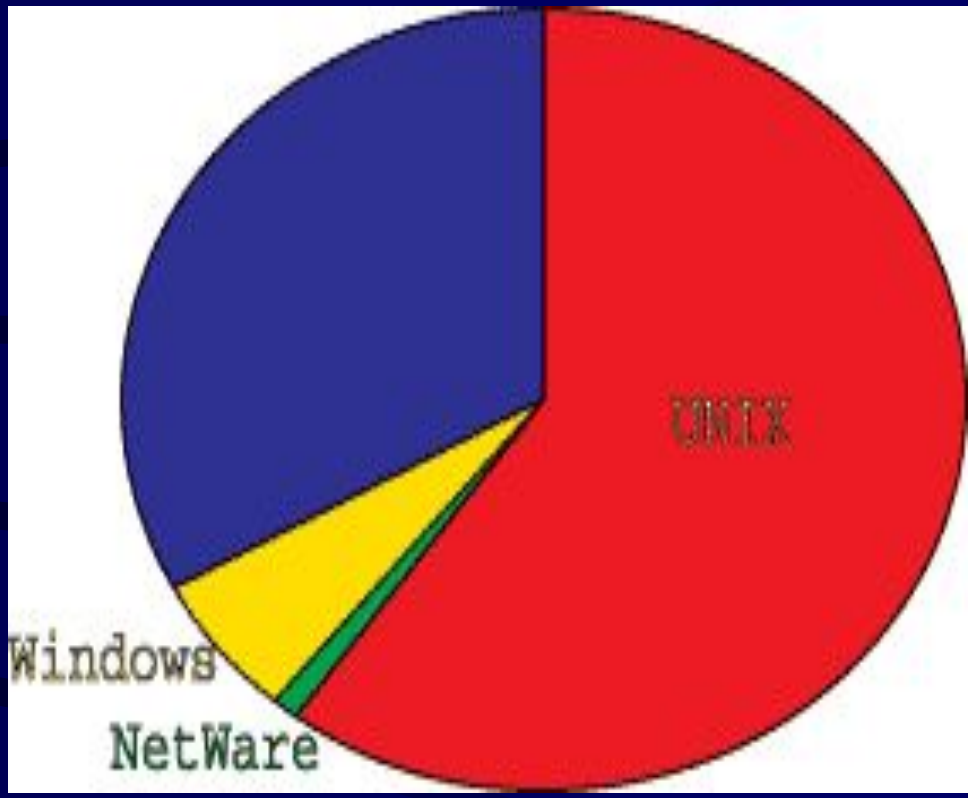
7.5. Тенденции на рынке операционных систем



Перераспределение рынка ОС по данным Gartner Group



Распределение от продажи серверов в ценовой категории менее 25 тыс. долл. по операционным системам (по данным Gartner Group, 2001 г.)



Распределение от продажи серверов в ценовой категории более 25 тыс. долл. по операционным системам (по данным Gartner Group, 2001 г.)

Последние новости от IDC о рынке операционных систем.

Доли рынка серверов выглядят следующим образом (анализ данных за прошлый год):

Windows: 55,1%

Linux: 23,1%

UNIX: 11%

NetWare: 9,9%

Доли рынка настольных (клиентских) машин (опять же данные за прошлый год):

Windows: 93,8%

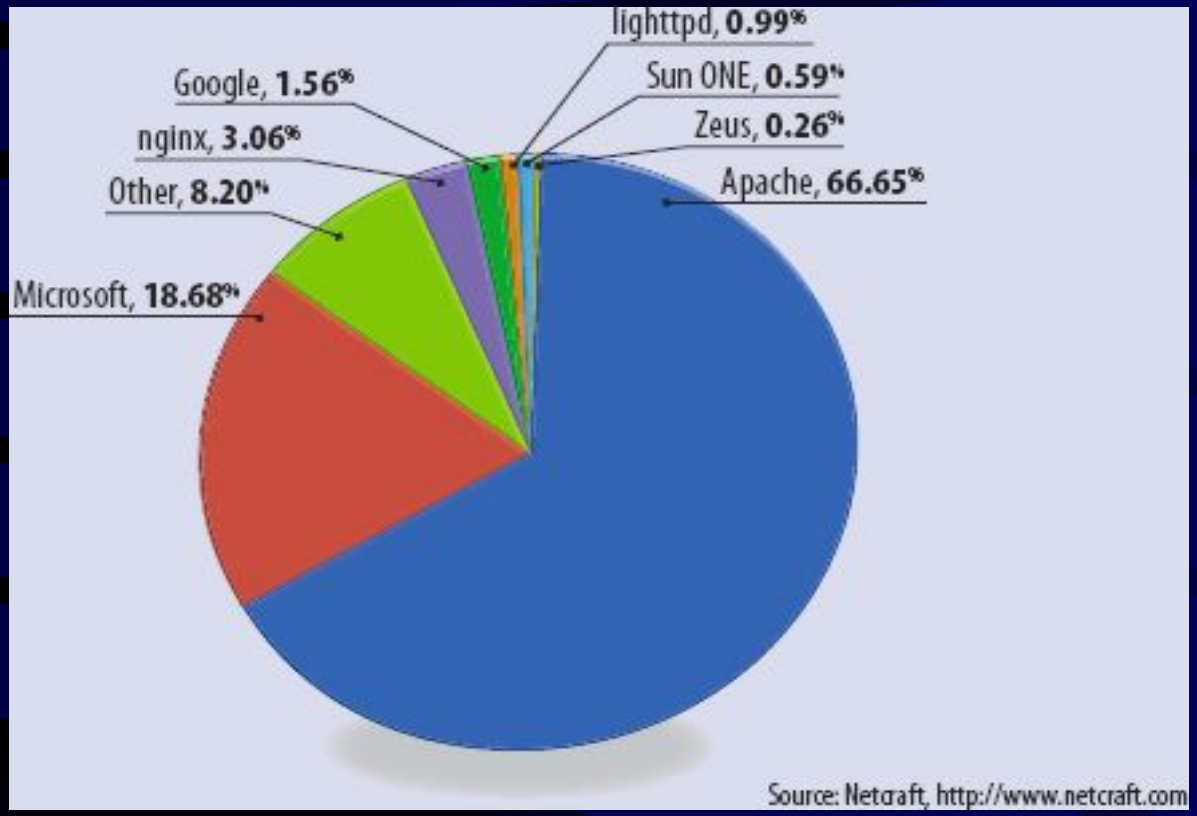
MacOS: 2,9%

Linux: 2,8%

В качестве гостевых ОС, в среде которых работают отдельные виртуальные машины, чаще всего (в двух случаях из трех) используются различные версии (32- и 64-разрядные) Windows (см. рисунок). Хотя, по мнению экспертов IDC, применение Linux увеличивается, доля этой ОС составляет пока лишь 12,7%. Заметная доля принадлежит мэйнфреймам и Unix, с которых в свое время и начиналась реализация идеи виртуальных машин. Так или иначе, но можно констатировать, что в подавляющем числе случаев использование VMS связано с компьютерами архитектуры x86.

системы

Изменяющийся облик open source. George Lawton. The Changing Face of Open Source, IEEE Computer, V. 42, No 5, May 2009



На программном обеспечении с открытыми кодами Apache работает гораздо больше крупнейших в мире Web-сайтов, чем на каком-либо другом программном обеспечении Web-серверов.

Неспособность компаний-поставщиков зарабатывать деньги при использовании подхода open source в чистом виде вынуждает многие из них изменять свою бизнес-модель. Теперь они считают открытые коды лишь одним из эффективных способов развития программного обеспечения, а не общественным движением.

7.6. Операционная система UNIX

7.6.1. История создания

1. Создание CTSS (Compatible Time Sharing System) в МТИ (1961 г.).
2. Создание MULTICS (Multiplexed Information and Computer Service), язык EOL (PL/1), МТИ + Bell Labs + General Electric, 1963 г.
3. Разработка усеченного варианта MULTICS на ассемблере для PDP-7 (UNICS – Uniplexed Information and Computer Service) – Кен Томпсон (1.01.1970).
4. Создание языка высокого уровня B (упрощение BCPL наследника CPL) и переработка Unix на этом языке – Томпсон.
5. Создание языка C (наследник B) – Ритчи.
6. Переписывание UNIX на C – Томпсон и Ритчи.
7. Статья Томпсона и Ритчи об ОС UNIX, 1974 г.
8. Версия 6 UNIX – 8200 строк C + 900 строк ассемблера –1974 г.
9. Первая переносимая версия UNIX (версия 7) –18000 строк C + 2110 строк ассемблера –1976 г.
10. Выпуск коммерческой версии UNIX фирмой AT&T (System III) – 1984 г., а затем UNIX System V.
11. Развитие UNIX Калифорнийским университетом в Беркли – 1BSD (Fist Berkeley Software Distribution), затем 2BSD, 3BSD, 4BSD.
12. Широкое распространение UNIX – Xenix, Minix, AIX, Sun OS, Solaris, Lnux.

7.6.2. Общая характеристика системы UNIX

ОС UNIX – интерактивная система, разработанная программистами и для программистов. Основные требования: простота, элегантность, непротиворечивость, мощь и гибкость.

Общие черты Unix независимо от версии:

1. Многопользовательский режим со средствами защиты от несанкционированных пользователей.
2. Реализация мультипрограммной работы в режиме разделения времени, основанная на использовании алгоритмов вытесняющей многозадачности.
3. Использование механизмов виртуальной памяти и свопинга для повышения уровня мультипрограммирования.
4. Унификация ввода-вывода на основе расширенного использования понятия файл.
5. Иерархическая файловая система, образующая единое дерево каталогов независимо от количества физических устройств, используемых для размещения файлов.
6. Переносимость системы за счет написания ее основной части на языке С.
7. Разнообразные средства взаимодействия процессов, в том числе через сеть.
8. Кэширование дисков для уменьшения среднего времени доступа к файлам.

7.6.3. Интерфейс системы UNIX

7.6.4. Структура ядра системы Unix

7.6.5. Оболочка системы UNIX

Система поддерживает графическое окружение X Windows, но многие программисты предпочитают интерфейс командной строки, создавая множество консольных окон и действуя так, как если бы у них было несколько алфавитно-цифровых терминалов, на каждом из которых работала бы оболочка (shell).

Существует много различных оболочек: sh, ksh, bash и др. После запуска оболочка печатает на экране символ приглашения к вводу (% или $) и ждет, когда пользователь введет командную строку.

Примеры командных строк:

1) cp file1 file2 (копировать файл file1, копия – file2)
2) head –20 file (печатать первые 20 строк файла file)
3) sort < in > out (программе sort взять в качестве входного файла in и направить вывод в файл out)
4) sort < in > temp; head -30 < temp; rm temp
5) sort < in | head -30 (канал)
6) sort < x | head & (фоновый процесс)

Файлы, содержащие команды оболочки,называются *сценариями оболочки. В них можно использовать конструкции if, for, while, case.*

7.6.6. Утилиты системы Unix

Кроме оболочки пользовательский интерфейс содержит большое число обслуживающих программ (утилит): 1. Программы (команды) управления файлами и каталогами. 2. Фильтры. 3. Средства разработки программ (текстовые редакторы, компиляторы). 4. Текстовые процессоры. 5. Системное администрирование. 6. Разное.

Программа	Функция
cat	Конкатенация нескольких файлов в стандартный выходной поток
chmod	Изменение режима защиты файла
cp	Копирование файла
cut	Вырезание колонок текста
grep	Поиск определенной последовательности символов в файле
head	Извлечение из файла первых строк
is	Распечатка каталога
make	Компиляция файла для создания двоичного файла
mkdir	Создание каталога
paste	Вставка колонок текста в файл
pr	Форматирование файла для печати
rm	Удаление файла
rmdir	Удаление каталога
sort	Сортировка строк файла по алфавиту
tail	Извлечение из файла последних строк
tr	Преобразование символа из одного набора в другой

7.6.7. Процессы в системе Unix

У каждого пользователя системы может быть одновременно несколько активных процессов, кроме того существуют десятки фоновых процессов (демонов). Типичный демон – ***cron daemon,*** предназначенный для планирования и запуска процессов. Системный вызов ***fork*** создает точную копию исходного процесса, называемого **родительским процессом**. Новый процесс называется дочерним. У процессов собственные образы памяти. Открытые файлы используются совместно.

```
pid = fork ( ); /* если fork завершился успешно, pid > 0 в родит. процессе */
if (pid < 0) {
handle_error ( ); /* fork потерпел неудачу (например, память переполнена)*/
} else if (pid > 0) {
                    /* здесь располагается родительская программа */
} else {
                    /* здесь располагается дочерняя программа */
}
```

Процессы взаимодействуют с помощью каналов. Синхронизация процессов достигается путем блокировки процесса при попытке прочитать данные из пустого канала. Например, когда оболочка выполняет строку **sort < f | head** она создаст два процесса, **sort** и **head,** и устанавливает между ними канал. Если канал переполняется, система приостанавливает работу **sort**, пока **head** не удалит хоть сколько-нибудь данных. **Процессы могут взаимодействовать также при помощи программных прерываний посылкой сигналов**.

Для управления процессами используются системные вызовы.

Системный вызов	Описание
pid = fork ( )	Создать дочерний процесс, идентичный родительскому
pid = waitpid (pid, &statloc, opts)	Ждать завершения дочернего процесса
s = execve (name, argv, envp)	Заменить образ памяти процесса
exit (status)	Завершить выполнение процесса и вернуть статус
s = sigaction (sig, &act, &oldact)	Определить действие, выполняемое при приходе сигнала
s = sigreturn (&context)	Вернуть управление после обработки сигнала
s =sigprocmask (how, &set, &old)	Исследовать или изменить маску сигнала
s = sigpending (set)	Получить или установить блокированные сигналы s =
sigsushtnd (sigmask)	Заменить маску сигнала и приостановить процесс
s = kill (pid, sig)	Послать сигнал процессу
s = pause ( )	Приостановить выполнение процесса до след. сигнала

```
while  (TRUE)  {                              /*Вечный цикл */
      type_prompt  ( );                       /*Вывести приглашение ко вводу*/
      read_command (command, params);  /*Прочитать с клавиатуры строку */
      pid = fork ( );                          /*Ответвить дочерний процесс*/
      if  (pid  < 0)  {
           printf (“Создать процесс невозможно”); /*Ошибка */
           continue;                           /*Следующий цикл */
       }
        if  (pid  != 0)  {
               waitpid ( -1, &status, 0 );          /*Род. пр-с  ждет завершения  доч.  пр-са*/
         }  else  {
         execve (command, params, 0);          /* Дочерний процесс выполняет работу*/
         }
}
```

Системные вызовы для управления потоками

Стандартом POSIX (P1003.1c) предусматривается реализация потоков в пространстве пользователя и ядра ОС. Наиболее часто применяемые вызовы управления потоками приведены в таблице. При системной реализации потоков они являются настоящими системными вызовами. При использовании потоков на уровне пользователя они полностью реализуются в динамической библиотеке в пространстве пользователя.

Системный вызов	Описание
pthread_create	Создать поток в адр. простр. вызывающего процесса
pthread_exit	Завершить поток
pthread_ join	Подождать пока не завершится процесс
pthread_mutex_init	Создать новый мьютекс
pthread_ mutex_destroy	Уничтожить мьютекс
pthread_ mutex_lock	Заблокировать мьютекс
pthread_ mutex_unlock	Разблокировать мьютекс
pthread_ cond_init	Создать условную переменную
pthread_ cond_ destroy	Уничтожить условную переменную
pthread_ cond_ wait	Ждать условную переменную
pthread_ cond_ signal	Разблокировать один поток, ждущий условную переменную

err = pthread_create (&tid, attr, function, arg); - этот вызов создает новый поток, в котором работает программа function, а arg передается программе в качестве параметра.

7.6.8. Реализация процессов в системе Unix

Ядро поддерживает две ключевые структуры данных, относящиеся к процессам: **таблицу процессов** (резидентная) и **структуру пользователя** (выгружается на диск, когда процесс отсутствует в памяти).

Таблица процессов содержит:

1. **Параметры планирования**. Приоритеты процессов, процессорное время, потребленное за последний учитываемый период, количество времени, проведенное процессом в режиме ожидания.
2. **Образ памяти**. Указатели на сегменты программы, данных и стека или на таблицы страниц. Когда процесса нет в памяти здесь содержится информация о его месте на диске.
3. **Сигналы.** Маски, характеризующие сигналы (игнорирование, перехват, блокирование)
4. **Разное.** Текущее состояние процесса, ожидаемые события, PID процесса, идентификатор пользователя и др.

Структура пользователя включает:

1. **Машинные регистры.**
2. **Информацию о текущем системном вызове (параметры и результаты).**
3. **Таблицу дескрипторов файлов.**
4. **Учетную информацию. Данные о процессорном времени, использованном в пользовательском и системном режимах.**
5. **Стек ядра для использования процессом в режиме ядра.**

Планирование в системеUNIX

1. **Низкоуровневый алгоритм** выбирает процесс, готовый к работе из очереди, имеющей высший приоритет (у процессов ядра – отрицательный, у процессов пользователя – положительный). Время кванта – 100 мс.

Priority = CPU_usage + nice + base

CPU_usage - “тики” таймера, при которых работал процесс;
CPU_usage (i) = CPU_usage (i – 1) / 2;
nice = от – 20 до + 20 (по умолчанию = 0);
base –назначается ОС (прошиты жестко и отрицательны для свопинга, дискового ввода-вывода и др.)

Priority пересчитывается каждую секунду.

2. **Высокоуровневый алгоритм** перемещает процессы из памяти на диск и обратно.

7.7. Операционная система Windows 2000

7.7.1. История создания

1. Появление фирмы MICROSOFT и интерпретатора языка BASIC (1981 г.) для микропроцессора Intel 8088.
2. Первый ПК IBM PC и MS DOS 1.0 (1981 г.), PC AT и MS DOS 3.0 (1984 г.).
3. Проект Lisa (графический интерфейс GUI Xerox, ПК Apple, С. Джобс, 1983 г.).

4. Оболочка MS DOS – Windows 1.0 (1985 г.).
5. Версия Windows 2.0 для PC AT (1987 г.).
6. Версия Windows 3.0 для ПК с Intel 386 (1990 г.).
7. Версии Windows 3.1 и 3.11 для ПК с Intel 386, 486 (1992 - 1994 гг.).
8. Windows 95 с большинством особенностей монолитной ОС на основе MS DOS 7.0, содержащая в значительной части 16-разрядный код (1995 г.).
9. Windows 98 со значительным наследием MS DOS, содержащая частично 16-разрядный код, и ориентацией на работу в Интернет (1998 г.).
10. Windows ME, в основе повторяющая Windows 98, но с возможностью восстановления настроек ПК при неверной установке параметров (2000 г.).

11. Полностью 32-разрядная операционная систем Windows NT 3.1 (1993 г.).
12. Значительно усовершенствованная Windows NT 4.0 (1996 г.).
13. Windows NT 5.0 - Windows 2000

Таблица 1-1. *Выпуски операционной системы Windows*

Название продукта	Номер версии	Дата выпуска
Windows NT 3.1	3.1	Июль 1993 г.
Windows NT 3.5	3.5	Сентябрь 1994 г.
Windows NT 3.51	3.51	Май 1995 г.
Windows NT 4.0	4.0	Июль 1996 г.
Windows 2000	5.0	Декабрь 1999 г.
Windows XP	5.1	Август 2001 г.
Windows Server 2003	5.2	Март 2003 г.

Windows Vista	6	2006	клиентская ОС
Windows Server 2008	6	2008	серверная ОС
Windows 7	7	2009	

Инструментальные средства MS для продвинутых пользователей (наборы утилит для отладки и мониторинга системы):

1. Support Tools - средства поддержки;
2. Software Development Kit (SDK) – средства разработки программных продуктов;
3. Driver Development Kit (DDK) – средства разработки драйверов;
4. Resource Kit – набор ресурсов.

Инструментарий поддержки распространяется на компакт-диске Windows 2000 (каталог \support\tools).

SDK и DDK можно получить на сайте msdn.microsoft.com.

Resource Kit распространяется как розничный продукт MS.

Мощный набор инструментов можно получить на сайте www.sysinternals.com.

7.7.2. Архитектура Windows

ОС Windows можно разделить на 2 части:

1. Основная часть ОС, работающая в режиме ядра (управление процессами, памятью, файловой системой , устройствами и т. д.).

2. Подсистемы окружения (среды), работающие в режиме пользователя (процессы, помогающие пользователям выполнять определенные системные функции).

Основная часть разделена на несколько уровней, каждый из которых пользуется службами лежащего ниже уровня. Основными уровнями являются:

- системные службы (сервисные процессы, являющиеся системными демонами);

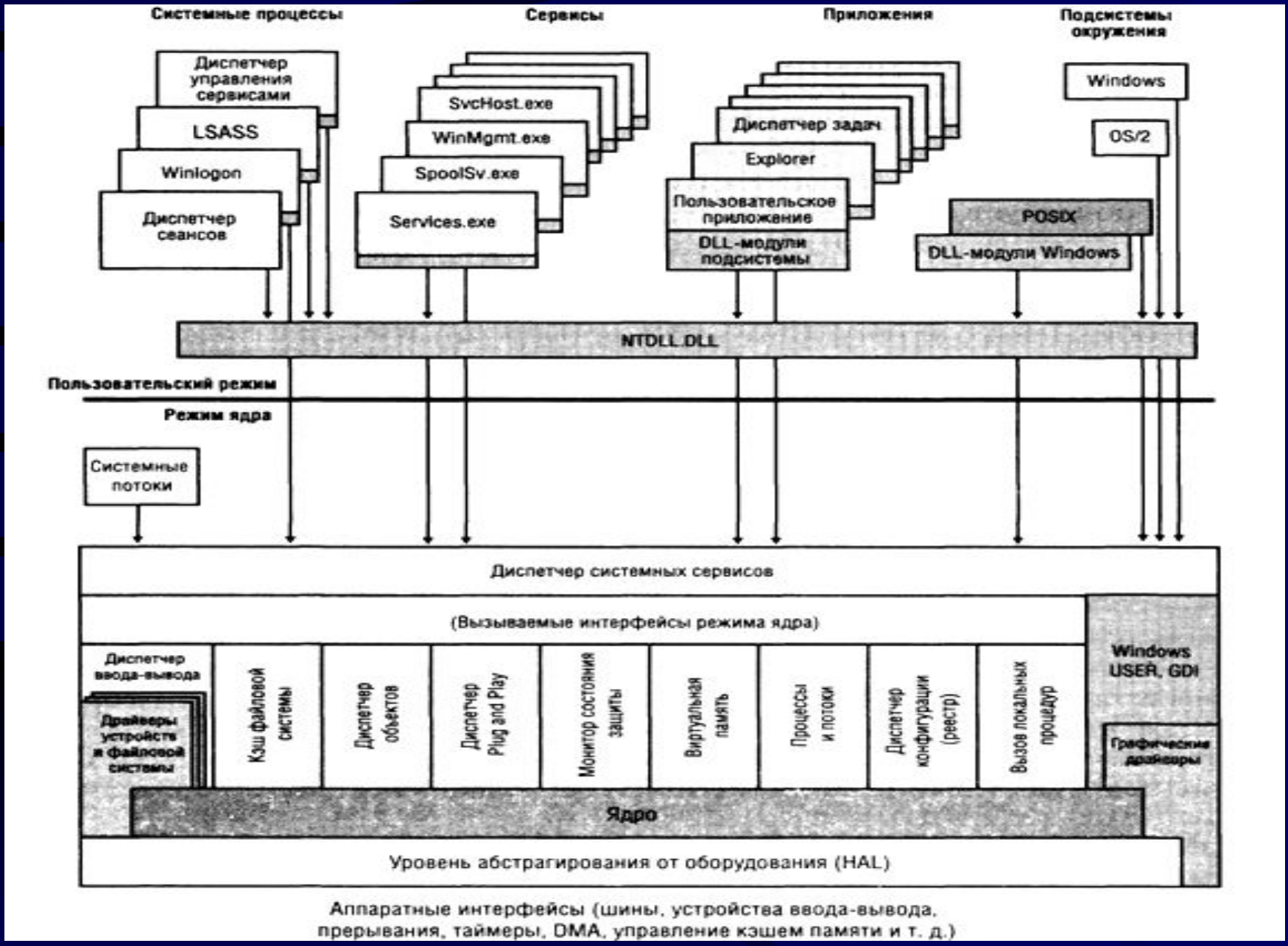
- исполняющая система (супервизор или диспетчер);

- драйверы устройств;

- ядро операционной системы;

- уровень аппаратных абстракций (HAL).

Два нижних уровня написаны на языке С и ассемблере и являются частично машинно-зависимыми. Верхние уровни написаны исключительно на языке С и почти полностью машинно-независимы. Драйверы написаны на С и в некоторых случаях на С++.

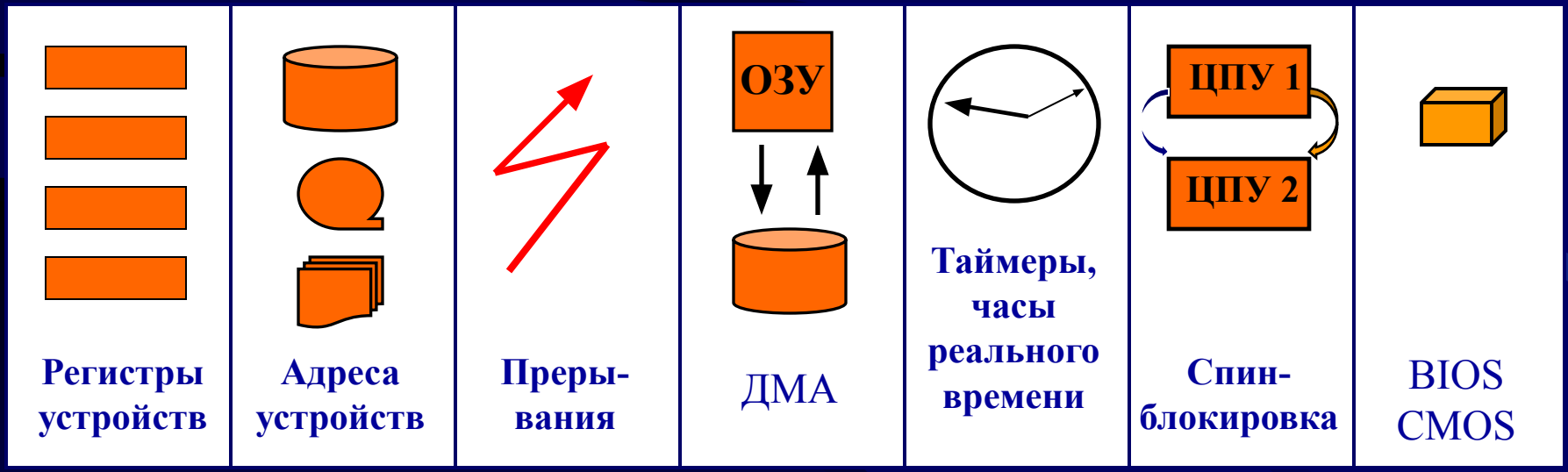

Системные процессы
Сервисы
Приложения
Подсистемы окружения
Диспетчер управления сервисами
LSASS
Winlogon
Диспетчер сеансов
SvcHost.exe
WinMgmt.exe
SpoolSv.exe
Services.exe
Диспетчер задач
Explorer
Пользовательское приложение
DLL-модули подсистемы
Windows
OS/2
POSIX
DLL-модули Windows
NTDLL.DLL
Пользовательский режим
Режим ядра
Системные потоки
Диспетчер системных сервисов
(Вызываемые интерфейсы режима ядра)
Диспетчер ввода-вывода
Драйверы устройств и файловой системы
Кэш файловой системы
Диспетчер объектов
Диспетчер Plug and Play
Монитор состояния защиты
Виртуальная память
Процессы и потоки
Диспетчер конфигурации (реестр)
Вызов локальных процедур
Windows USER, GDI
Графические драйверы
Ядро
Уровень абстрагирования от оборудования (HAL)
Аппаратные интерфейсы (шины, устройства ввода-вывода, прерывания, таймеры, DMA, управление кэшем памяти и т. д.)

Уровень аппаратных абстракций (Hardware Abstraction Layer – HAL)

Работа уровня HAL заключается в том, чтобы предоставить всей остальной системе абстрактные аппаратные устройства, свободные от индивидуальных особенностей аппаратуры. Эти устройства представляются в виде машинно-независимых служб (процедурных вызовов и макросов), которые могут использоваться остальной ОС и драйверами.

В уровень HAL включены те службы, которые зависят от от набора микросхем материнской платы и меняются от машины к машине в разумных предсказуемых пределах.

Некоторые функции уровня HAL (HAL.DLL)



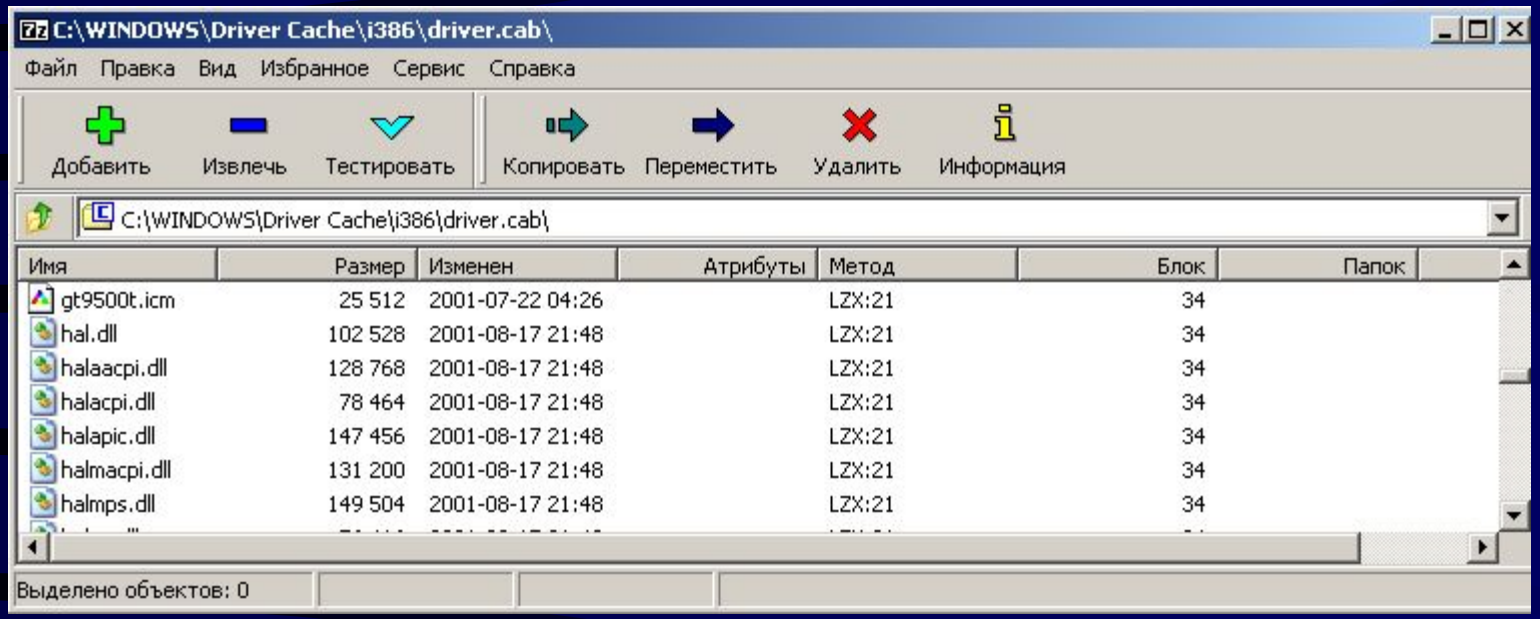


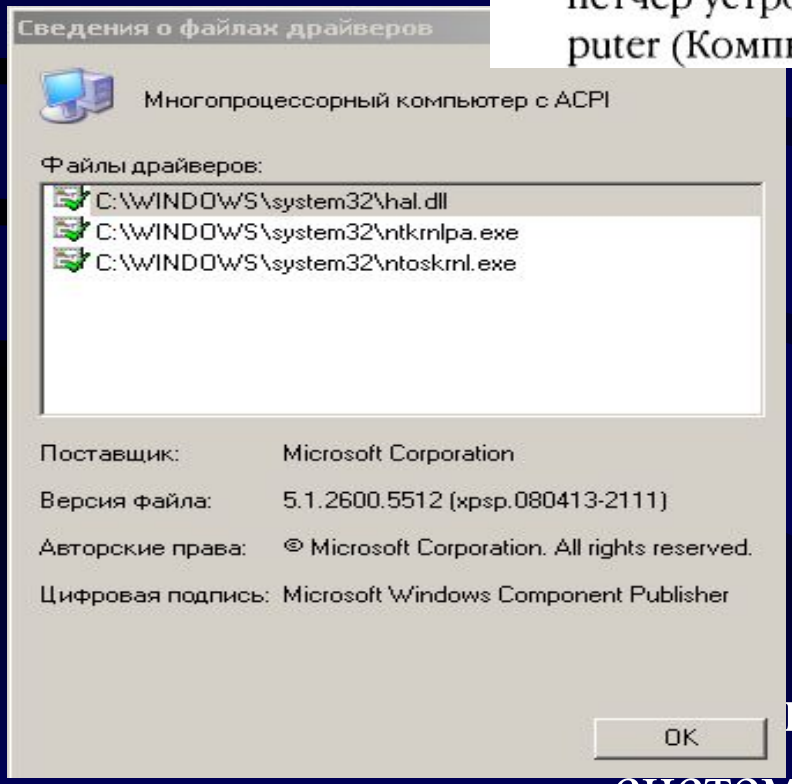
Таблица 2-6. *Список модулей HAL для x86 в \Windows\Driver\Cache\i386\Driver.cab*

Имя файла HAL	Поддерживаемые системы
Hal.dll	Стандартные персональные компьютеры (ПК)
Halacpi.dll	ПК с ACPI (Advanced Configuration and Power Interface)
Halapic.dll	ПК с APIC (Advanced Programmable Interrupt Controller)
Halaacpi.dll	ПК с APIC и ACPI
Halmps.dll	Многопроцессорные ПК
Halmacpi.dll	Многопроцессорные ПК с ACPI
Halborg.dll	Рабочие станции Silicon Graphics (только для Windows 2000; больше не продаются)
Halsp.dll	Compaq SystemPro (только для Windows XP)

ПРИМЕЧАНИЕ В базовой системе Windows Server 2003 нет HAL, специфических для конкретных вендоров.

Определить, какой модуль HAL используется на вашей машине, можно двумя способами.

1. Откройте файл \Windows\Repair\Setup.log, найдите строку с *Hal.dll*. Имя файла, стоящее в этой строке после знака равенства, соответствует имени модуля HAL, извлеченного из Driver.cab с дистрибутивного носителя.
2. Откройте Device Manager (Диспетчер устройств): щелкните правой кнопкой мыши значок My Computer (Мой компьютер) на рабочем столе, выберите команду Properties (Свойства), откройте вкладку Hardware (Оборудование) и щелкните кнопку Device Manager (Диспетчер устройств). Проверьте имя «драйвера» для устройства Computer (Компьютер).



Уровень ядра

Назначение ядра – сделать остальную часть ОС независимой от аппаратуры. Для этого ядро на основе низкоуровневых служб HAL формирует абстракции более высоких уровней. Например, у уровня HAL есть вызовы для связывания процедур обработки прерываний с прерываниями и установки их приоритетов. Больше ничего в этом отношении HAL не делает. Ядро же предоставляет полный механизм для переключения контекста и планирования потоков.

Ядро также предоставляет низкоуровневую поддержку двум классам объектов ядра – управляющим объектам и объектам диспетчеризации. Эти объекты используются системой и приложениями для управления ресурсами компьютерной системы: процессами, потоками , файлами и т. д.

Каждый объект ядра – это блок памяти, выделенный ядром, доступный только ему и представляющий собой структуру данных, в которой содержится информация об объекте.

К управляющим объектам ядра относятся: объекты заданий, процессов, потоков, прерываний, DPC (Deferred Procedure Call – отложенный вызов процедуры), APC (Asynchronous Procedure Call – асинхронный вызов процедуры)

К объектам диспетчеризации ядра относятся объекты, изменение состояния которых могут ждать потоки. Это –семафоры, мьютексы, события, таймеры, очереди, файлы, порты, маркеры доступа и др.

Исполняющая система

Написана на языке С, не зависит от архитектуры машины и относительно просто может быть перенесена на новые машины.Исполняющая система состоит из 10 компонентов, между которыми нет жестких границ. Компоненты одного уровня могут вызывать друг друга. Большинство компонентов представляют собой процедуры, которые выполняются потоками системы в режиме ядра.

Менеджер объектов управляет всеми объектами, создаваемыми или известными операционной системе (процессами, потоками, семафорами, файлами и т. д.). Он управляет пространством имен, в которое помещается созданный объект, чтобы к нему можно было обратиться по имени. Остальные компоненты ОС пользуются объектами во время своей работы.

Менеджер ввода-вывода предоставляет остальной части ОС независимый от устройств ввод-вывод, вызывая для выполнения физического ввода-вывода соответствующий драйвер.

Менеджер процессов управляет процессами и потоками, включая их создание и завершение. Он основывается на объектах потоков и процессов ядра и добавляет к ним дополнительные функции. Это ключевой элемент многозадачности.

Менеджер памяти реализует архитектуру виртуальной памяти со страничной подкачкой по требованию ОС. Он управляет преобразованием виртуальных страниц в физические и реализует правила защиты, ограничивающие доступ каждому процессу только теми страницами, которые принадлежат его адресному пространству.

Менеджер безопасности приводит в исполнение сложный механизм безопасности Windows 2000, удовлетворяющий требованиям класса C2 Оранжевой книги Министерства обороны США.

Менеджер кэша хранит в памяти блоки диска, которые использовались последнее время, чтобы ускорить доступ к ним и обслуживает все файловые системы. Взаимодействует с менеджером виртуальной памяти, чтобы обеспечить требуемую непротиворечивость.

Менеджер plug-and-play управляет установкой новых устройств, контролирует их динамическое подключение и осуществляет при необходимости загрузку драйверов.

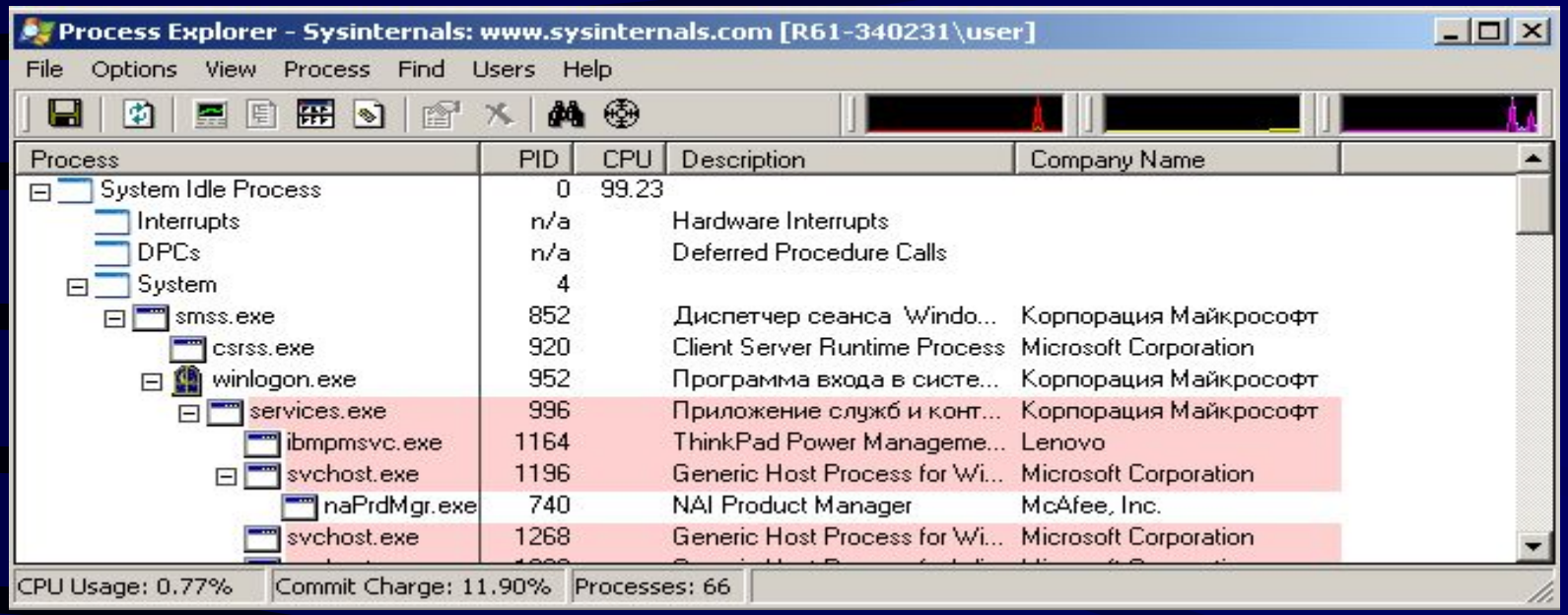
Менеджер энергопотребления управляет потреблением энергии, следит за состоянием батарей, взаимодействует с ИБП.

Менеджер конфигурации отвечает за сохранение реестра.

Менеджер вызова локальной процедуры обеспечивает высокоэффективное взаимодействие между процессами и их подсистемами. Поскольку этот путь нужен для выполнения некоторых системных вызовов, эффективность оказывается критичной, поэтому здесь не используются стандартные механизмы межпроцессного взаимодействия.

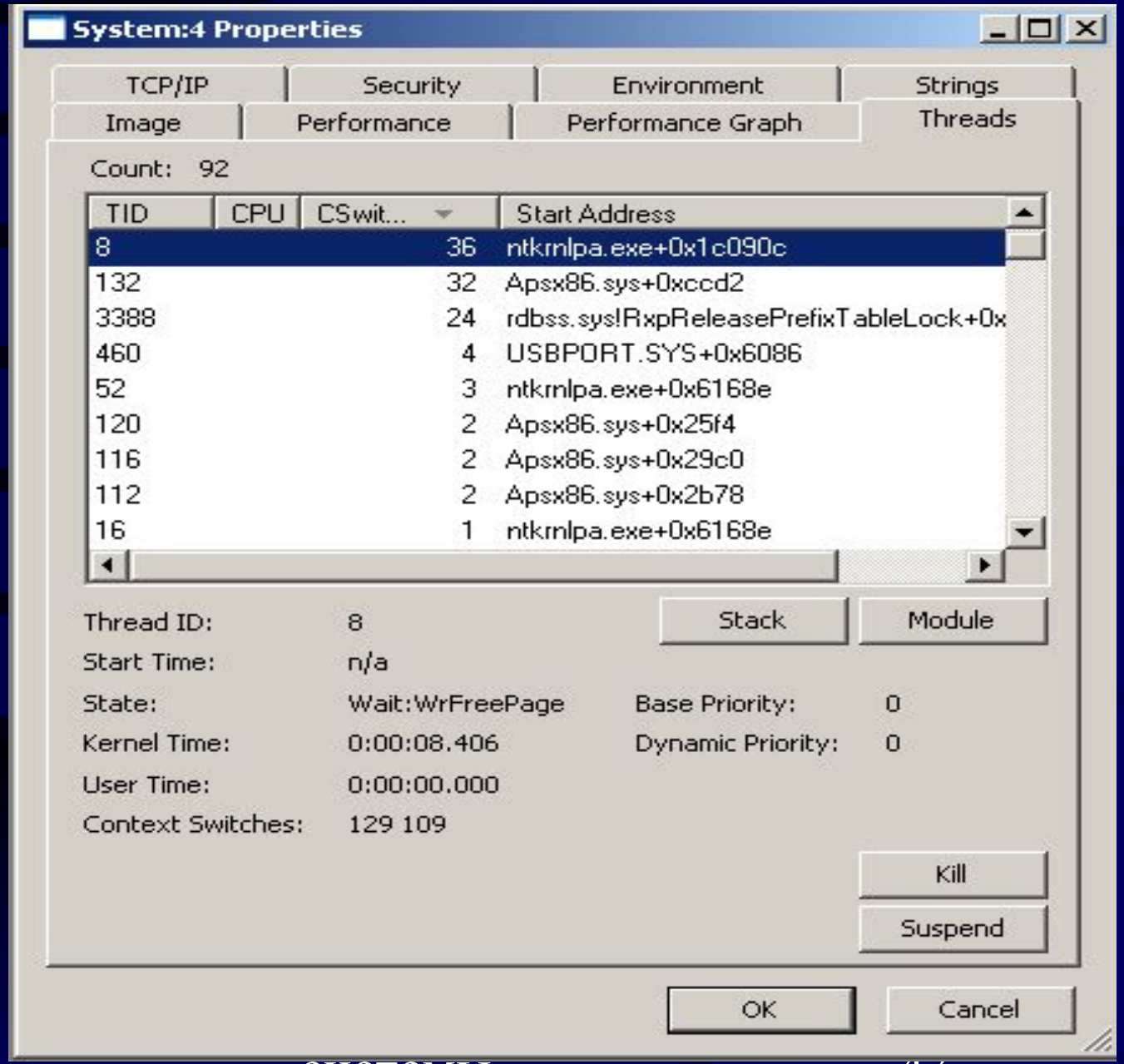
Интерфейс графических устройств GDI (Graphic Device Interface) управляет графическими изображениями для монитора и принтеров, предоставляя системные вызовы для пользовательских программ. (Интерфейс Win32 и модуль GDI превосходят по объему всю остальную исполняющую систему.)

Системные процессы и службы предоставляют интерфейс к исполняющей системе. Они принимают системные вызовы Windows 2000 и вызывают необходимые части исполняющей системы для их выполнения.



В каждой системе Windows выполняются перечисленные ниже процессы. (Два из них, Idle и System, не являются процессами в строгом смысле этого слова, поскольку они не выполняют какой-либо код пользовательского режима.)

- Процесс Idle (включает по одному потоку на процессор для учета времени простоя процессора).
- Процесс System (содержит большинство системных потоков режима ядра).
- Диспетчер сеансов (Smss.exe).
- Подсистема Windows (Csrss.exe).
- Процесс входа в систему (Winlogon.exe).
- Диспетчер управления сервисами (Services.exe) и создаваемые им дочерние процессы сервисов (например, универсальный процесс для хостинга сервисов, Svchost.exe).
- Серверный процесс локальной аутентификации (Lsass.exe).


System:4 Properties
TCP/IP
Security
Environment
Strings
Image
Performance
Performance Graph
Threads
Count: 92
TID
CPU
CSwit...
Start Address
8 36 ntkrnlpa.exe+0x1c090c
132 32 Apsx86.sys+0xccd2
3388 24 rdbss.sys!RxpReleasePrefixTableLock+0x
460 4 USBPORT.SYS+0x6086
52 3 ntkrnlpa.exe+0x6168e
120 2 Apsx86.sys+0x25f4
116 2 Apsx86.sys+0x29c0
112 2 Apsx86.sys+0x2b78
16 1 ntkrnlpa.exe+0x6168e
Thread ID: 8
Start Time: n/a
State: Wait:WrFreePage
Kernel Time: 0:00:08.406
User Time: 0:00:00.000
Context Switches: 129 109
Base Priority: 0
Dynamic Priority: 0
Stack
Module
Kill
Suspend
OK
Cancel

Драйверы устройств устанавливаются в систему, добавляются в реестр и затем динамически загружаются при каждой загрузке системы. Драйверы не являются частью файла ntoskrnl.exe.

Основная часть ОС, состоящая из ядра и исполняющей системы, хранится в файле **ntoskrnl.exe (2140 Кбайт).** Уровень **HAL**, представляющий собой библиотеку общего доступа, находится в файле **hal.dll (102 Кбайт).** Интерфейс Win32 и графических устройств хранятся в файле **win32.sys (1690 Кбайт)**. Системный интерфейс для связи пользовательского режима с режимом ядра – файл **NTDLL.DLL** содержит **694 Кбайт**.

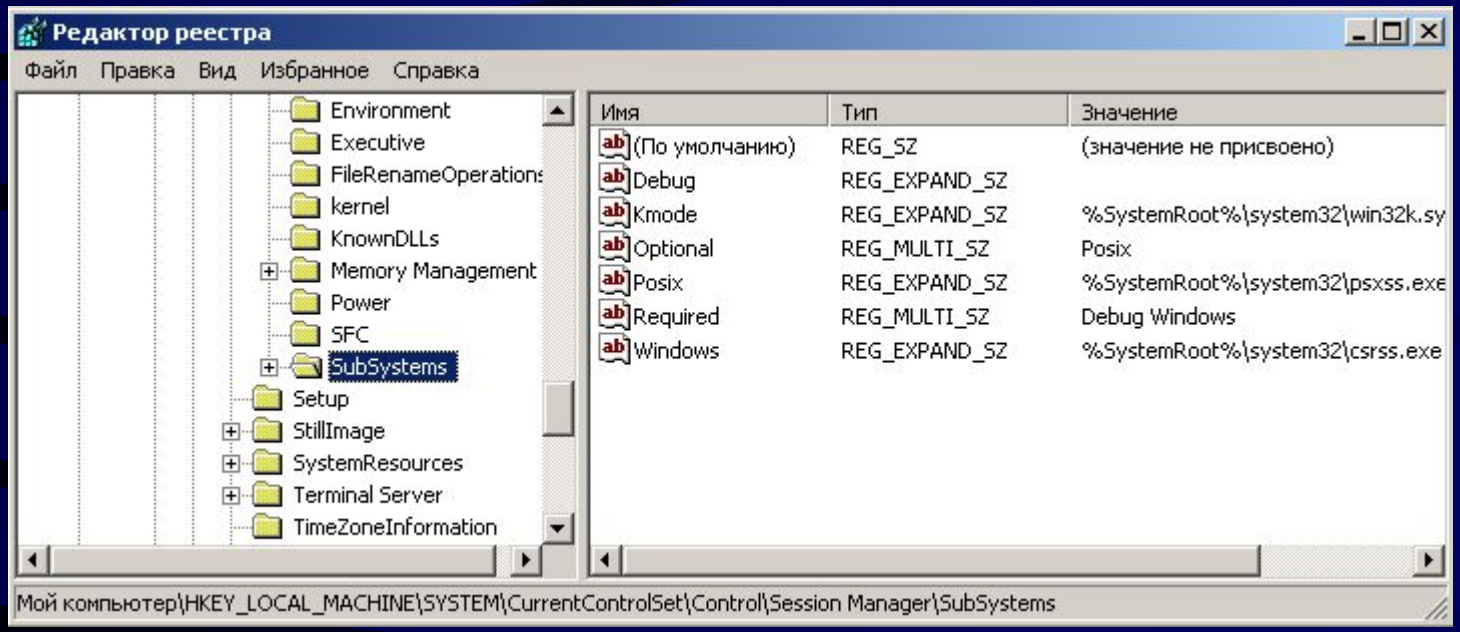
Подсистемы окружения

В Windows существует три типа компонентов, работающих в режиме пользователя: динамические библиотеки DLL, подсистемы окружения и служебные процессы.Эти компоненты работают вместе, предоставляя пользовательским процессам три различных документированных интерфейса прикладного программирования API (**Win32, POSIX, OS/2).**

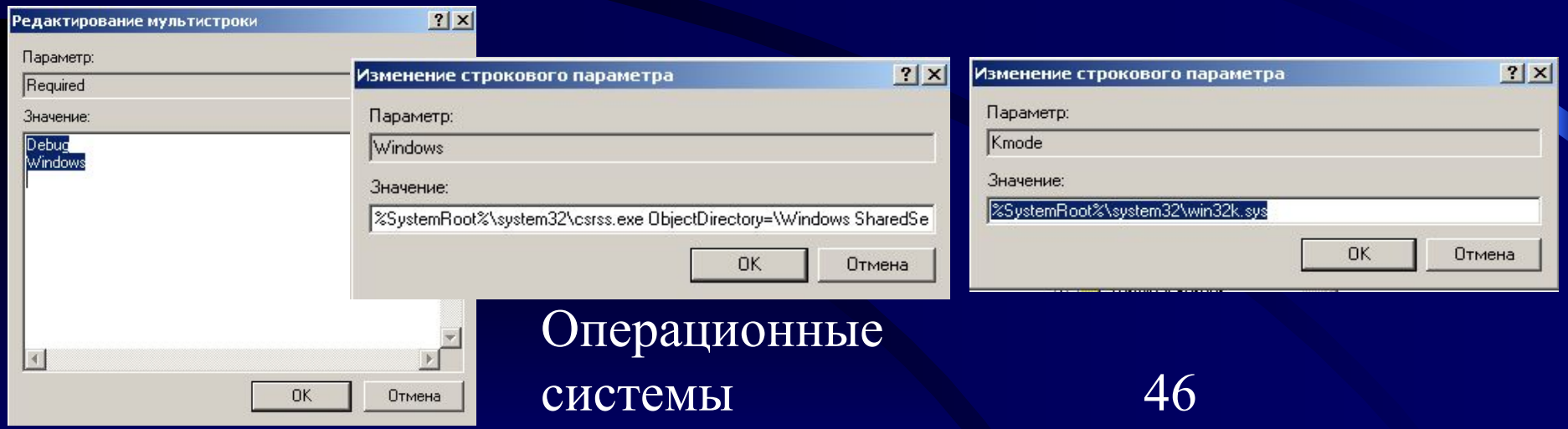
У каждого интерфейса есть список библиотечных вызовов, которые используются программистами. Работа библиотек DLL и подсистем окружения заключается в том, чтобы реализовать функциональные возможности опубликованного интерфейса, скрывая истинный интерфейс системных вызовов от прикладных программ.

Программы, пользующиеся интерфейсом Win32, содержит, как правило, большое количество обращений к функциям Win32 API. Поэтому, если работает одновременно несколько программ, то в памяти будут находиться многочисленные копии одних и тех же библиотечных процедур. Чтобы избежать подобной проблемы Wndows поддерживает динамически присоединяемые библиотеки DLL. При этом при запуске нескольких приложений, использующих одну и ту же DLL, в памяти будет находиться только одна копия текста DLL.

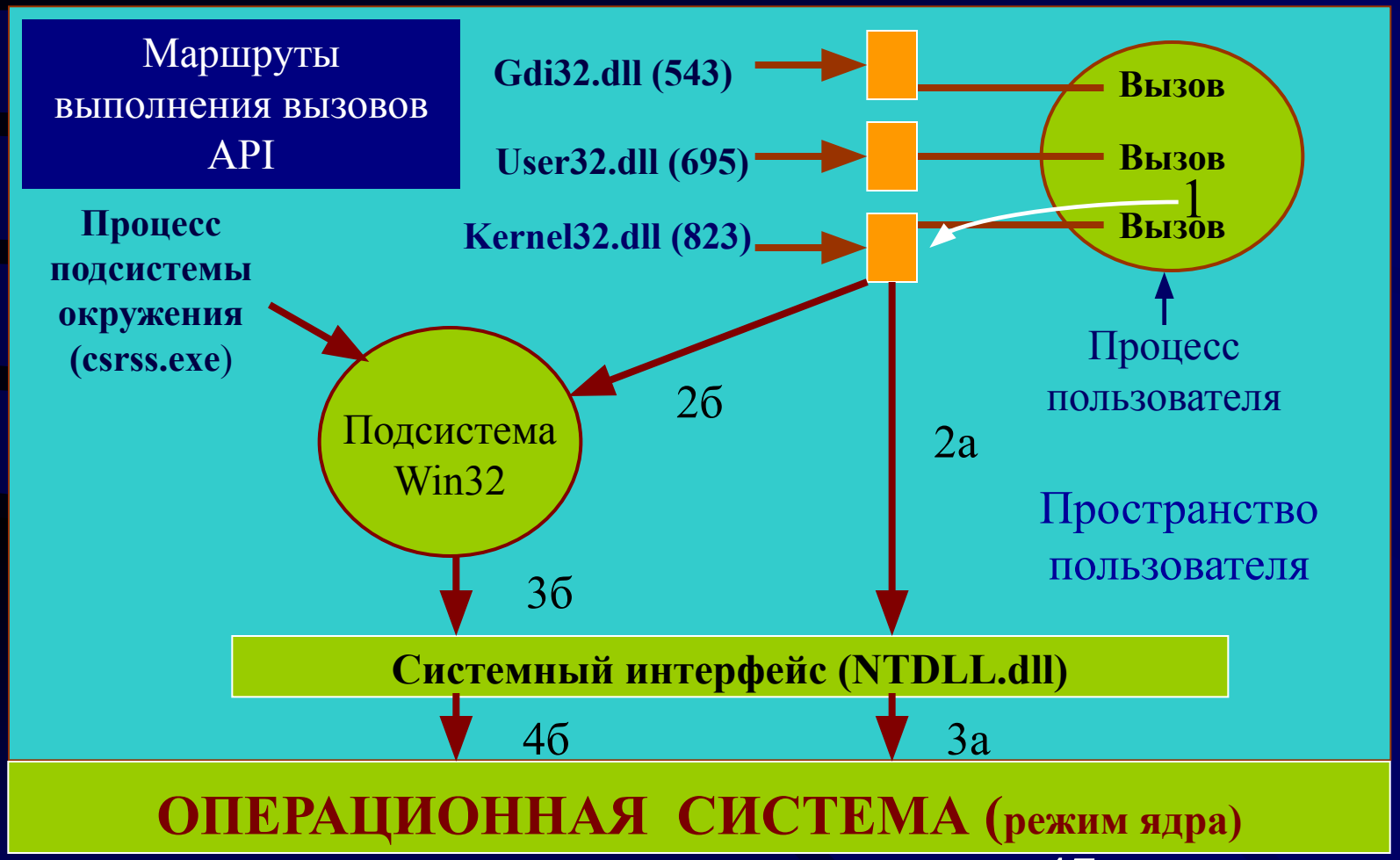
В каталоге \winnt\system32 есть более 800 отдельных файлов DLL (130 Мбайт при числе вызовов API более13000).



Подсистема Windows работает всегда (обеспечивает клавиатуру, мышь, экран), остальные подсистемы запускаются опционально (Optional). Параметр Required определяет список подсистем, загружаемых при запуске ОС. В параметре Windows указывается спецификация файла Csrss. Параметр Kmode содержит имя файла подсистемы Windows, работающего в режиме ядра.



Пользовательские приложения вызывают системные сервисы через DLL подсистем.
1. Функция реализована в пользовательском режиме внутри DLL подсистемы.
2. DLL – библиотека обращается к другой DLL (NTDLL.dll), которая обращается к ядру ОС (2а-3а).
3. Для выполнения функции происходит обращение к подсистеме Win32, которая выполняет всю работу самостоятельно или обращается к системному вызову (2б-3б-4б).



7.7.5. Файловая система Windows 2000

Основные свойства файловой системы NTFS:

1. Поддержка больших файлов и больших дисков (объем до 2^{64} байт).
2. Восстанавливаемость после сбоев и отказов программ и аппаратуры управления дисками.
3. Высокая скорость операций, в том числе для больших дисков.
4. Низкий уровень фрагментации, в том числе для больших дисков.
5. Гибкая структура, допускающая развитие за счет добавления новых типов записей и атрибутов файлов с сохранением совместимости с предыдущими версиями ФС.
6. Устойчивость к отказам дисковых накопителей.
7. Поддержка длинных символьных имен.
8. Контроль доступа к каталогам и отдельным файлам.

Файл NTFS – не просто линейная последовательность байтов, характерная для FAT-систем и Unix, а множество атрибутов, представляемых в виде потока байтов. Файл имеет несколько коротких потоков (имя, идентификатор и др.) и один или несколько длинных потоков с данными (ff:stream1, ff:stream2 и др.).

Структура тома NTFS

Основой структуры тома является *главная таблица файлов* (***Master File Table, MFT***), которая содержит одну или несколько записей для каждого файла тома и одну запись для самой себя (размер записи – 1, 2 или 4 Кбайт).

Том состоит из последовательности кластеров, порядковый номер кластера в томе – ***логический номер кластера*** *(Logical Cluster Number, LCN).*

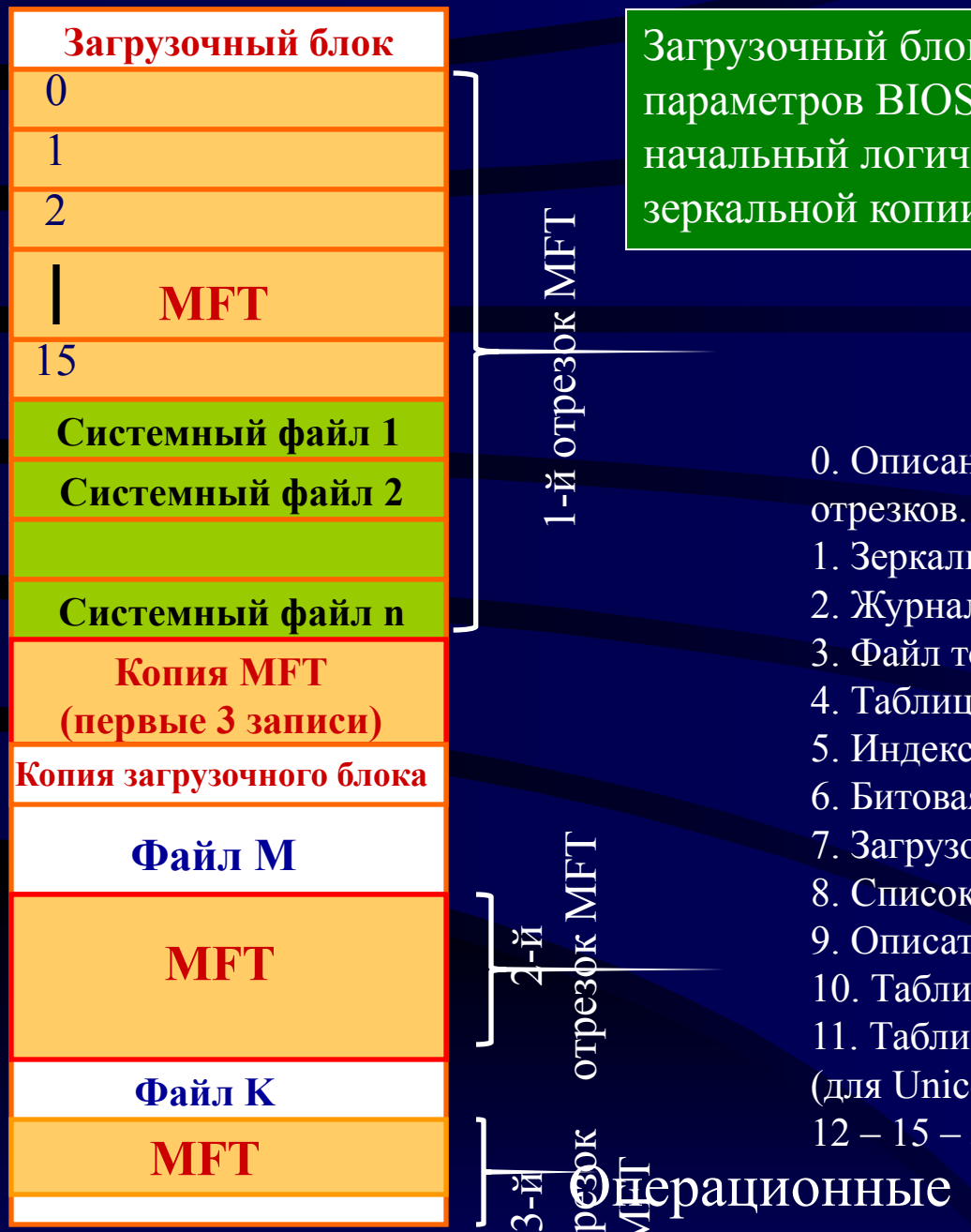
Файл состоит из последовательности кластеров, порядковый номер кластера внутри файла называется ***виртуальным номером кластера*** (*Virtual Cluster Number, VCN*). Размер кластера от 512 байт до 64 Кбайт.

Базовая единица распределения дискового пространства – ***отрезок*** – непрерывная область кластеров.

Адрес отрезка – (LCN, k), k –количество кластеров в отрезке.

Адрес файла (или его части) – (LCN, VCN, k).

Файл целиком размещается в записи таблицы MFT (если позволяет размер). В противном случае в записи MFT хранится резидентная часть файла (некоторые его атрибуты), а остальная часть файла хранится в отдельном отрезке тома или нескольких отрезках.



Загрузочный блок содержит стандартный блок параметров BIOS, количество блоков в томе, начальный логический номер кластера основной и зеркальной копии MFT.

Файлы метаданных

0. Описание MFT, в том числе адреса всех ее отрезков.
1. Зеркальная копия MFT.
2. Журнал для восстановления файловой системы.
3. Файл тома (имя, версия и др. информация).
4. Таблица определения атрибутов.
5. Индекс корневого каталога.
6. Битовая карта кластеров.
7. Загрузочный сектор раздела.
8. Список дефектных кластеров.
9. Описатели защиты файлов.
10. Таблица квот.
11. Таблица преобразования регистра символов (для Unicode).

12 – 15 – зарезервировано.

Структура файлов NTFS

Файлы и каталоги состоят из набора атрибутов. Каждая запись MFT состоит из заголовка, за которым следуют атрибуты. Атрибуты содержат следующие поля: тип, длина, имя (образуют заголовок) и значение.

Атрибуты, используемые в записях MFT:

1. Стандартная информация (сведения о владельце, флаговые биты, время создания, время обновления и др.).
2. Имя файла в кодировке Unicode, м.б. повторено для имени MS DOS.
3. Список атрибутов (содержит ссылки на номера записей MFT, где расположены атрибуты), используется для больших файлов.
4. Версия – номер последней версии файла.
5. Дескриптор безопасности – список прав доступа ACL.
6. Версия тома –используется в системных файлах тома.
7. Имя тома.
8. Битовая карта MFT – карта использования блоков тома.
9. Корневой индекс – используется для поиска файлов в каталоге.
10. Размещение индекса – нерезидентная часть индексного списка (для больших файлов).
11. Идентификатор объекта – 64-разрядный идентификатор файла, уникальный для данного тома.
12. Данные файла.
13. Точка повторного анализа (монтирование и симв. ссылки)

Файлы NTFS в зависимости от способа размещения делятся на небольшие, большие, очень большие и сверхбольшие.

Пример небольшого файла NTFS

Блоки диска 20 – 23, 64 – 65, 80 - 82

Пример большого файла NTFS