ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ
ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ИНСТИТУТ
РАДИОТЕХНИКИ, ЭЛЕКТРОНИКИ И АВТОМАТИКИ
(ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ)»

КАФЕДРА ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

РЕФЕРАТ

по предмету

«Операционные системы»

«Процесс загрузки операционных систем Windows семейства NT: Windows 2000, XP, 2003 Server»

Выполнен
студентом факультета ВМС
группы ВВ-2-06
Котоминым Иваном

МОСКВА 2008

Оглавление

Опишем процесс загрузки Windows с момента установки и рассмотрим весь процесс выполнения загрузочных файлов. Драйверы устройств являются важной частью процесса загрузки, поэтому мы объясним, в какой момент они загружаются и инициализируются. Затем мы рассмотрим, как инициализируется система исполнения программ и как ядро запускает часть Windows, исполняющуюся в режиме пользователя через запуск процесса Менеджера сессий, подсистемы Windows, и процесса входа в систему Winlogon.

Предварительная загрузка

Процесс загрузки Windows не начинается непосредственно после включения питания компьютера или нажатия кнопки перезагрузки. Очевидно, что для этого сначала должна быть произведена установка Windows. А во время установки, в определенный момент происходит подготовка системного диска.

Со времен MS-DOS существует стандарт разбиения физических дисков на тома. Операционные системы разбивают диски на отдельные области, называемые разделами, и используют файловые системы для форматирования разделов в тома. Жесткий диск может содержать до четырех первичных разделов. Но так как подобный принцип разбиения ограничивает количество томов до четырех, используется специальный тип раздела – расширенный раздел, который также позволяет выделить до четырех разделов. Расширенные разделы могут содержать другие расширенные разделы, и так далее, делая число томов, которые можно разместить на жестком диске, практически бесконечным. На рис.1 приведен пример разбиения на разделы.

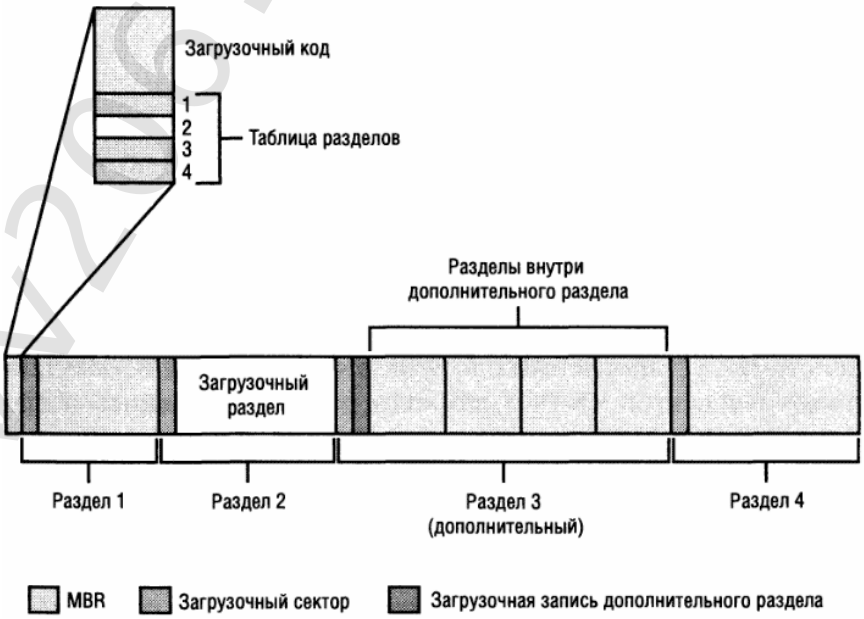

Рис.1. Пример структуры разделов.

Единицей адресации физических дисков является *сектор*. Обычно он составляет 512 байт. Программы, подготавливающие жесткие диски к разбиению на разделы, такие как Fdisk или программа установки Windows, записывают сектор с данными, называемый главной загрузочной записью (Master Boot Record), в первый сектор жесткого диска. Главная загрузочная запись включает в себя определенный набор инструкций, называемых загрузочным кодом, и таблицу, называемую таблицей разделов, с четырьмя записями, определяющими расположение первичных разделов на диске.

При загрузке компьютера, первым исполняется код из BIOS, «зашитый» в ROM компьютера. BIOS выбирает загрузочное устройство, считывает с него MBR в память, и передает управление загрузочному коду из нее.

Код, содержащийся в главной загрузочной записи, вне зависимости от создавшей его операционной системы, выполняет сходные функции. Во-первых, этот код сканирует первичную таблицу разделов в поисках определенного флага, означающего то, что соответствующий раздел является загрузочным. Когда MBR находит хотя бы один такой флаг, она считывает первый сектор помеченного раздела в память и передает управление коду этого раздела. Такой тип раздела называется *загрузочным*, а первый сектор этого раздела – *загрузочным сектором*. Том, определенный на загрузочном разделе, называется *системным томом*.

Операционная система обычно записывает загрузочный сектор на диск без участия пользователя. Например, когда программа установки Windows записывает MBR на диск, она заодно и записывает загрузочный сектор на первый загрузочный раздел диска. В случае, если программа установки опознает перезаписываемый сектор как загрузочный, созданный MS-DOS или Windows 95-Millenium, то она копирует его содержимое в файл Bootsect.dos в корневом каталоге раздела.

Перед записью загрузочного сектора раздела, программа установки удостоверяется в том, что этот раздел отформатирован в поддерживаемой файловой системе (FAT, FAT32 или NTFS) путем форматирования загрузочного раздела в файловой системе, указанной пользователем. После завершения форматирования загрузочного раздела, установка копирует используемые Windows файлы в созданный системный том.

Также частью выполняемой установкой работы является создание файла конфигурации загрузочного меню, Boot.ini, в корневом каталоге системного тома. Этот файл содержит параметры загрузки устанавливаемой ОС и любых других, уже существующих, установок. Если в Bootsect.dos был скопирован верный образ загрузочного сектора MS-DOS, то добавляется еще одна запись для ее загрузки.

Загрузочный сектор и Ntldr

Программа установки должна знать тип файловой системы раздела, в который происходит установка, так как содержимое загрузочного сектора различается в зависимости от существующей ФС. Например, если тип загрузочного раздела FAT, то Windows записывает загрузочный сектор, «понимающий» эту файловую систему. Роль кода загрузочного сектора состоит в том, чтобы предоставит Windows минимальную необходимую информацию о структуре и формате тома для считывания файла Ntldr из корневой папки тома. Поэтому загрузочный сектор содержит необходимый минимум кода для поддержки чтения файлов.

После того, как загрузочный сектор загружает Ntldr в память, он передает управление на его точку входа. Если Ntldr не будет обнаружен в корневой директории, будет отображено сообщение об ошибке "BOOT: Couldn’t find NTLDRP" для ФС FAT или "NTLDR is missing" для NTFS.

Ntldr начинает работу в то время, когда система работает в *реальном режиме*. Реальный режим – это такой режим работы процессора, когда не производится преобразования адресов памяти из виртуальных в физические, реальные. Это означает, что программы, оперирующие адресами памяти, воспринимают их как физические адреса, и поэтому программами адресуем только первый мегабайт физической памяти. В этом режиме функционируют простые программы MS-DOS.

Однако первым действием, предпринимаемым Ntldr, является перевод системы в *защищенный режим*. На этой стадии загрузки все еще не осуществляется преобразование адресов, но становятся доступными все 32 бита адреса памяти. После перевода в реальный режим, Ntldr имеет доступ ко всей физической памяти. После создания достаточного количества страниц памяти для получения доступа к первым 16 МБ с включенной подкачкой, происходит ее включение. Защищенный режим с включенной подкачкой явсяется режимом, в котором происходит обычная работа Windows.

После включения подкачки, Ntldr обретает полную функциональность. Тем не менее, он все еще опирается на функции, предоставляемые загрузочным кодом, для доступа к IDE дискам и экрану. Функции загрузочного кода временно отключают подкачку и переключают процессор в режим, в котором могут исполняться службы, предоставляемые BIOS. Затем Ntldr считывает файл Boot.ini из корневой папки, используя встроенный код для чтения файловой системы. Как и загрузочный сектор, Ntldr поддерживает только режим чтения FAT и NTFS, но умеет читать поддиректории.

Затем Ntldr очищает экран. Если в корне диска есть верный файл Hiberfil.sys, то процесс загрузки сокращается за счет считывания его содержимого в память и передаче управления коду ядра восстанавливаемой системы. Hiberfil.sys будет использоваться только в том случае, если при последнем завершении работы система была переведена в спящий режим.

Если в Boot.ini определен более чем один вариант загрузки, то пользователю предоставляется меню загрузки. Иначе же начинается загрузка системы, определенной единственным вариантом загрузки.

Если запись в Boot.ini ссылается на установку MS-DOS, то Ntldr считывает содержимое Bootsect.dos в память, переключается в 16-битный реальный режим, и передает управление коду из считанного файла. Таким образом, код Bootsect.dos выполняется, как будто он был считан напрямую из загрузочного сектора.

Записи в Boot.ini могут включать дополнительные аргументы, передаваемые ядру или другим компонентам загрузки. Такие аргументы могут определять параметры отладки, настройки памяти, дополнительные параметры обнаружения оборудования или работы процессора и т.д.

После выбора загружаемой системы, Ntldr загружает и запускает исполнение Ntdetect.com, 16-битной программы реального режима, использующей BIOS для определения аппаратного обеспечения компьютера и других параметров, включающих в себя:

- Время и дату по часам компьютера
- Типы шин (например, ISA, PCI, и т.д.) в системе и идентификаторы подключенных к ним устройств
- Число и емкость установленных в компьютере дисков
- Типы подключенных мышей
- Число и тип подключенных параллельных портов
- Типы видеоадаптеров, представленных в компьютере

Эта информация сохраняется в памяти в служебных структурах данных, которые впоследствии будут сохранены в реестре.

В Windows 2000, Ntldr очищает экран и выводит надпись “Starting Windows” и индикатор хода загрузки в виде полосы в нижней части экрана. Он остается на нуле до

начала загрузки загрузочных драйверов. Если в этот момент нажать F8, то появится расширенное загрузочное меню, позволяющее выбрать такие варианты загрузки, как последняя удачная конфигурация, безопасный режим, режим отладки и т.д. В Windows XP и 2003 Server отображается графический логотип-заставка, а не индикатор.

Если Ntldr запускается на 64-битной системе и в выбранной записи Boot.ini указано 64-битное ядро, то процессор переводится в *«длинный режим»*, в котором используется длина слова в 64 бита.

Далее, Ntldr начинает загрузку файлов, необходимых для инициализации ядра, с загрузочного тома. *Загрузочный том* – тот том, который соответствует разделу с системной папкой (обычно \Windows) загружаемой установки Windows. При этом процесс проходит через следующие этапы:

1. Загрузка соответствующих образов ядра и уровня аппаратных абстракций (HAL) – по умолчанию это файлы Ntoskrnl.exe и Hal.dll. Если эти файлы найти не удается, то выводится сообщение «Не удалось запустить Windows, так как следующие файлы повреждены или отсутствуют», а также имена файлов.

2. Считывает куст[1] SYSTEM, расположенный в \Windows\System32\Config\System, для определения необходимых для загрузки драйверов.

3. Просматривает копию в памяти куста SYSTEM и выделяет драйвера загрузочных устройств. Эти драйвера являются необходимыми для завершения загрузки. В реестре у них установлен параметр запуска в «0», т.е. первоочередной.

4. Добавляет драйвер файловой системы, отвечающий за операции с той системой, в которой оформлен установочный раздел, в список с загрузочными драйверами. Драйвер ФС обязан быть загружен именно на этом этапе; иначе, ядру бы потребовались драйвера для их же загрузки, что бы привело к круговой зависимости.

5. Загружает драйверы, обязательные для запуска системы. Ход загрузки отражается индикатором «Starting Windows». Если в Boot.ini указан параметр /SOS, то вместо индикатора Ntldr выводит имя файла каждого загрузочного драйвера. На этом этапе драйверы лишь загружаются, а их инициализация происходит позже.

6. Подготавливает регистры процессора для выполнения Ntoskrnl.exe.

На этом участие Ntldr в процессе загрузки заканчивается. Для инициализации системы Ntldr вызывает главную функцию из Ntoskrnl.exe.

[1] Куст – файл, содержащий ветвь реестра.

Инициализация ядра и компонентов исполнительной системы

Вызывая Ntoskrnl.exe, Ntldr передает структуру данных с копией строки из Boot.ini (представляющей выбранный вариант загрузки), с указателем на таблицы памяти (сгенерированные Ntldr для описания физической памяти в данной системе), а также с указателем на загруженные в память копии кустов реестра HARDWARE и SYSTEM и на список загруженных драйверов.

Ntoskrnl начинает первую из двух фаз процесса инициализации (они нумеруются от 0). Большинство компонентов исполнительной системы имеет инициализирующую функцию, которая принимает параметр, определяющий текущую фазу.

В фазе 0 прерывания отключены. Предназначение этой фазы в том, чтобы сформировать рудиментарные структуры, необходимые для вызова сервисов в фазе 1. Главная функция Ntoskrnl вызывает функцию инициализации системы, которая в свою очередь вызывает процедуры инициализации процессора и ядра операционной системы для каждого процессора. Работая на стартовом процессоре, процедура инициализации процессора выполняет создание всех внутренних структур данных, разделяемых всеми процессорами. Затем каждый экземпляр инициализации ядра ОС вызывает функцию, отвечающую за управление фазой 0.

Фаза 0 начинается с вызова HAL-функции инициализации системы, позволяющей HAL взять управление этим процессом на себя. Одной из ее задач является подготовка системного контроллера прерываний каждого процессора к обработке прерываний и конфигурирование таймера, используемого для учета распределяемого процессорного времени.

Далее вызываются процедуры инициализации для диспетчеров памяти, объектов, процессов и Plug and Play, а также монитора состояния защиты. Эти компоненты выполняют следующие инициализирующие операции:

1. Диспетчер памяти формирует таблицы страниц и внутренние структуры данных, необходимые для предоставления базовых сервисов, связанных с памятью. Кроме того, он создает и резервирует пространство для кэша файловой системы, а также выделяет области для пулов подкачиваемой и неподкачиваемой памяти.

2. При инициализации диспетчера объектов определяются объекты, необходимые для создания его пространства имен, чтобы другие компоненты могли помещать в него свои объекты. Также создается таблица дескрипторов для поддержки учета ресурсов.

3. Монитор состояния защиты инициализирует объект типа «маркер доступа» и

использует его для подготовки первого маркера, назначаемого начальному процессу.

4. Диспетчер процессов производит большую часть своей инициализации в фазе 0, определяя типы объектов «процесс» и «поток» и создавая списки для отслеживания активных процессов и потоков. Он также создает объект «процесс» для начального процесса и присваивает ему имя Idle. Наконец, диспетчер процессов создает процесс System и системный поток для выполнения процедуры второй стадии инициализации. Этот поток не запускается сразу после создания, поскольку прерывания пока запрещены.

5. Далее наступает фаза 0 в инициализации диспетчера Plug and Play, в ходе которой просто инициализируется ресурс системы, используемый для синхронизации ресурсов шин.

Когда на каждом процессоре управление возвращается к функции инициализации ядра Windows, она передает его циклу Idle. В результате поток System начинает фазу 1. Дополнительные процессоры все еще ждут начала своей инициализации.

В фазе 1 выполняются следующие операции:

1. Для подготовки системы к приему прерываний от устройств и для разрешения прерываний вызывается HalInitSystem.

2. Вызывается загрузочный видеодрайвер (\Windows\System32\Bootvid.dll), который выводит экран-заставку, показываемую в процессе запуска Windows. (В Windows XP и Windows Server 2003 этот драйвер отображает ту картинку, которую Ntldr ранее вывел на экран.)

3. Инициализируется диспетчер электропитания.

4. Инициализируются системные часы, текущее значение которых сохраняется как время загрузки системы.

5. В многопроцессорной системе инициализируются остальные процессоры, и начинается выполнение команд.

6. Диспетчер объектов создает корневой каталог пространства имен и каталог сопоставлений DOS-имен устройств (\Global?? в Windows XP и Windows Server 2003).

7. Вызывается исполнительная система для создания своих типов объектов, необходимых для синхронизации процессов и их планировщика.

8. Ядро инициализирует структуры данных планировщика процессов.

9. Монитор состояния защиты создает в пространстве имен диспетчера объектов

каталог \Security и инициализирует структуры данных аудита (если аудит системы разрешен).

10. Для создания системных рабочих потоков вызывается диспетчер памяти.

11. Загружаются таблицы поддержки национальных языков.

12. Загружается Ntdll.dll.

13. Диспетчер кэша инициализирует структуры данных кэша файловой системы и создает свои рабочие потоки.

14. Диспетчер конфигурации создает в пространстве имен объект «раздел реестра» \Registry и копирует переданные Ntldr начальные данные в кусты реестра HARDWARE и SYSTEM.

15. Инициализируются структуры данных драйвера файловой системы.

16. Диспетчер Plug and Play вызывает PnP BIOS.

17. Наступает момент инициализации диспетчера ввода-вывода. Эта стадия запуска системы занимает 50% времени. Происходит последовательная загрузка драйверов устройств.

18. Диспетчер ввода-вывода прежде всего инициализирует различные внутренние структуры и создает типы объектов «устройство» и «драйвер». Затем он вызывает диспетчер Plug and Play, диспетчер электропитания и HAL, чтобы начать динамическое перечисление и инициализацию устройств.

19. После этого загружаются и инициализируются драйверы, необходимые для запуска системы.

20. Включается подкачка страниц для кода режима ядра (в Ntkrnl и драйверах).

21. Вызывается диспетчер электропитания для инициализации своих структур данных.

22. На завершающем этапе создается процесс Smss диспетчера сеансов. Smss отвечает за создание среды пользовательского режима, которая предоставляет визуальный интерфейс Windows.

Smss, Csrss и Winlogon

Smss похож на любой другой процесс пользовательского режима, но имеет два существенных отличия. Во-первых, Windows считает его доверяемой частью системы. Во-вторых, Smss является встроенным (native) приложением. Как доверяемый компонент Smss может выполнять операции, доступные лишь немногим процессам, например

создавать маркеры защиты. А как встроенное приложение Smss использует не Windows API, а базовые API-функции исполнительной системы, в совокупности называемые Windows Native API. Smss не обращается к Windows API, поскольку при его запуске подсистема Windows еще не функционирует. Запуск подсистемы Windows и является одной из его первых задач.

Затем Smss вызывает диспетчер конфигурации, который завершает инициализацию реестра, заполняя все его разделы.

Основной поток Smss выполняет следующие инициализирующие операции.

1. Создает два потока, ожидающие клиентские запросы (например, на загрузку новой подсистемы или на создание сеанса).
2. Определяет символьные ссылки на имена устройств MS-DOS.
3. Если установлены Terminal Services, создает в пространстве имен диспетчера объектов каталог \Sessions (для нескольких сеансов).
4. Запускает программы, указанные в параметрах реестра менеджера сессий. Как правило, в нем содержится одна команда на запуск проверки диска.
5. Выполняет отложенные действия по переименованию и удалению файлов.
6. Создает дополнительные страничные файлы.
7. Инициализирует реестр. Диспетчер конфигурации заполняет реестр, загружая кусты HKLM\SAM, HKLM\SECURITY и HKLM\SOFTWARE.
8. Создает системные переменные окружения.
9. Загружает часть подсистемы Windows, работающую в режиме ядра (Win32k.sys). Инициализирующий код в Win32k.sys использует видеодрайвер для переключения экрана в разрешение, определенное в профиле по умолчанию. Таким образом, в этот момент видеоадаптер переключается с VGA-режима, используемого загрузочным видеодрайвером, в выбранное для данной системы разрешение.
10. Запускает процессы подсистем, в том числе Csrss.
11. Запускает процесс Winlogon. Этапы запуска Winlogon кратко описываются ниже.

После выполнения вышеперечисленных операций основной поток Smss переходит к бесконечному ожиданию процессов Csrss и Winlogon. Поскольку от этих процессов зависит функционирование Windows, в случае их неожиданного завершения Smss

вызывает крах системы.

Далее Winlogon продолжает инициализацию, выполняя такие операции, как создание начального объекта «Оконная станция» и объектов рабочего стола. Затем Winlogon создает процесс диспетчера управления сервисами, который загружает все сервисы и драйверы устройств, помеченные для автоматического запуска, а также запускает процесс LSASS (подсистемы локальной аутентификации).

После того как SCM инициализирует автоматически запускаемые сервисы и драйверы устройств, а пользователь успешно зарегистрируется в системе, загрузка считается успешно завершенной.

Запустив SCM, Winlogon ждет уведомления об интерактивном входе. Получив такое уведомление, Winlogon загружает куст реестра из профиля зарегистрировавшегося пользователя и отображает его на HKCU. Затем запускается процесс инициализации - исполняемый файл, указанный в реестре (по умолчанию Userinit.exe).

Userinit.exe выполняет следующие операции:

1. Обрабатывает пользовательские и машинные сценарии входа.

2. Если политика группы задает какую-либо квоту в профиле пользователя, Userinit.exe ее примененяет.

3. Запускает оболочку (или оболочки), указанную в реестре (по умолчанию — Explorer.exe).

Заключение

Итак, после включения питания компьютера и до окончания загрузки система проходит множество этапов: обработку кода MBR, загрузочного сектора активного раздела, работу загрузчика, передающего контроль над загрузкой ядру, а также процесс разворачивания операционной системой необходимых структур данных и подготовка устройств ввода-вывода, а также запуск процессов, обеспечивающих разные стороны работы и безопасность операционной системы.

Литература

1. Mark E. Russinovich, David A. Solomon «Microsoft Windows Internals, Fourth Edition: Microsoft Windows Server 2003, Windows XP, and Windows 2000», Microsoft Press, 2004