ГЛАВА 4

Запуск и завершение работы системы

В этой главе описываются стадии загрузки Windows 2000, а также параметры, влияющие на процесс ее запуска. Далее поясняется, что происходит при корректном завершении работы системы. В заключение мы рассмотрим, какие причины могут вызывать крах Windows 2000 и что делать, если это произошло. Понимание тонкостей процесса загрузки поможет Вам в диагностике проблем, возникающих при загрузке Windows 2000.

Процесс загрузки

Описание процесса загрузки Windows 2000 мы начнем с рассмотрения установки Windows 2000, а затем исследуем работу модулей Ntldr и Ntdetect. Поскольку драйверы устройств играют ключевую роль в процессе загрузки, будет уделено внимание и тому, как они контролируют собственную загрузку и инициализацию. Далее мы поясним, как инициализируются компоненты исполнительной системы и как ядро запускает пользовательскую часть Windows 2000, активизируя процессы Session Manager (Smss.exe) и Winlogon, а также подсистему Win32. Попутно Вы узнаете, что происходит внутри системы на момент вывода тех или иных текстовых сообщений, появляющихся на экране в процессе загрузки. Компоненты, участвующие в процессе загрузки перечислены в таблице 4-1.

Таблица 4-1. *Компоненты, участвующие в процессе загрузки*

Компонент	Режим работы	Описание
Код главной загрузочной записи (master boot record, MBR)	16-разрядный реальный	Считывает в память загрузочные сектора раздела
Загрузочный сектор	16-разрядный реальный	Считывает корневой каталог для загрузки Ntldr
Ntldr	16-разрядный реальный и 32-разрядный защищенный (активизирует поддержку подкачки страниц)	Считывает Boot.ini, выводит загрузочное меню, загружает Ntoskrnl.exe, Bootvid.dll, Hal.dll и драйверы, необходимые для загрузки и последующего запуска системы

см. след. стр.

Таблица 4-1. *продолжение*

Компонент	Режим работы	Описание
Ntoskrnl.exe	32-разрядный защищенный с поддержкой подкачки страниц	Инициализирует компоненты исполнительной системы, драйверы, необходимые для загрузки и запуска системы, подготавливает систему к выполнению встроенных приложений и запускает Smss.exe
Smss	Встроенное 32-разрядное приложение	Загружает подсистему Win32, включая Win32k.sys и Csrss.exe, и запускает процесс Winlogon
Winlogon	Встроенное 32-разрядное приложение	Загружает SCM, подсистему локальной аутентификации (Lsass) и выводит на экран диалоговое окно для входа в систему
Диспетчер управления сервисами (service control manager, SCM)	Встроенное 32-разрядное приложение	Загружает и инициализирует автоматически запускаемые драйверы устройств и сервисы Win32

Что предшествует загрузке

Процесс загрузки Windows 2000 начинается не при включении компьютера или нажатии кнопки Reset, а еще при установке этой системы на компьютер. На одном из этапов работы программы установки Windows 2000 (Windows 2000 Setup) происходит подготовка основного жесткого диска системы: на нем размещается код, в дальнейшем участвующий в процессах загрузки. Прежде чем рассказывать, что делает этот код, мы покажем, как и в какой области жесткого диска он размещается.

Стандарт разбиения физических жестких дисков на тома существует в системах типа *x*86 со времен первых версий MS-DOS. Операционные системы Microsoft разбивают жесткие диски на дискретные области, называемые *разделами* (partitions). После форматирования с использованием файловых систем (типа FAT и NTFS) разделы образуют тома. На жестком диске может быть до четырех главных разделов (primary partitions). Поскольку это ограничило бы количество томов на одном диске, данная схема предусматривает особый тип раздела — *дополнительный* (extended partition), что дает до четырех дополнительных разделов в главном разделе. Дополнительные разделы могут содержать другие дополнительные разделы, те в свою очередь — третьи дополнительные разделы и т. д. Так что диск можно разбить практически на бесконечное число томов. Пример разбиения жесткого диска на разделы показан на рис. 4-1. (Подробнее о разбиении жестких дисков в Windows 2000 см. главу 10.)

Единицей адресации физических дисков является *сектор*. Типичный размер сектора жесткого диска на IBM-совместимом компьютере — 512 байт. Такие утилиты, как Fdisk в MS-DOS или программа Windows 2000 Setup, позволяющие создавать на жестком диске логические диски, записывают в первый сектор жесткого диска специальные данные, создавая таким образом главную

загрузочную запись (MBR) диска. Размер MBR фиксирован. Она состоит из набора машинных команд (загрузочный код) и таблицы разделов с четырьмя записями, которые определяют расположение главных разделов на диске. Первый код, выполняемый при загрузке IBM-совместимого компьютера, называется BIOS — он хранится в ПЗУ компьютера. BIOS считывает MBR в память и передает управление ее загрузочному коду.

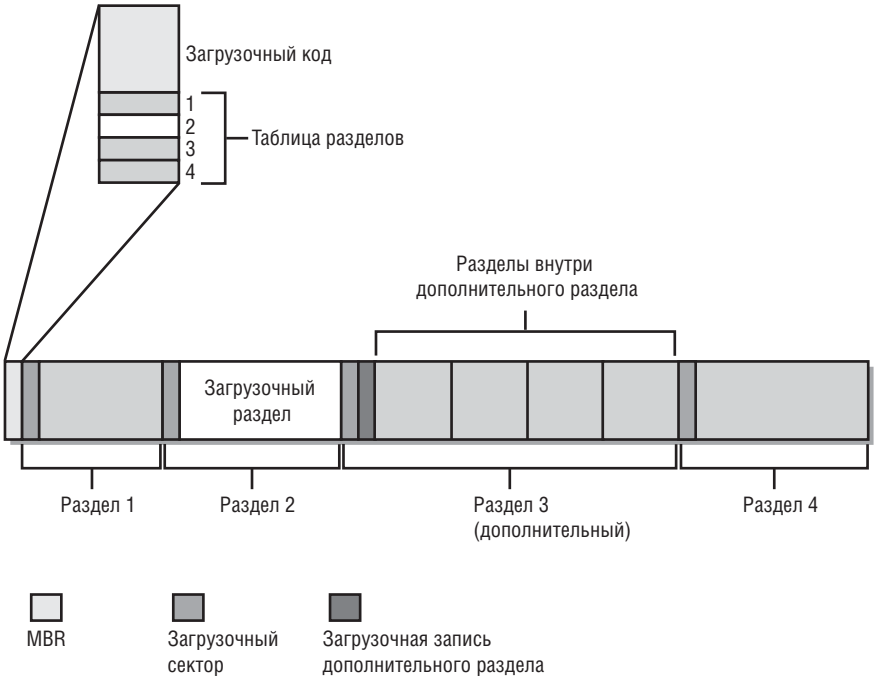

Рис. 4-1. *Пример структуры разделов жесткого диска*

MBR начинает со сканирования таблицы разделов в поисках раздела, помеченного особым флагом. Этот флаг сигнализирует, что данный раздел является загрузочным. Как только MBR обнаружит хотя бы один такой флаг, она считывает в память код из первого сектора раздела, помеченного флагом, и передает ему управление. Такой раздел называется *загрузочным* — как и его первый сектор.

Операционная система, как правило, ведет запись в загрузочные сектора без участия пользователя. Например, программа установки Windows 2000 при записи MBR одновременно создает в первом загрузочном разделе жесткого диска свой загрузочный сектор. Перед этим программа установки проверяет, является ли он сейчас загрузочным сектором MS-DOS. Если да, Setup сначала копирует содержимое загрузочного сектора в файл Bootsect.dos, помещая его в корневой каталог раздела.

Перед записью в загрузочный сектор Windows 2000 Setup проверяет совместимость текущей файловой системы этого раздела с Windows 2000. В любом случае она может отформатировать данный раздел с использованием выбранной Вами файловой системы (FAT, FAT32 или NTFS). Если раздел уже отфор-

матирован, Вы можете пропустить этот этап. После того как загрузочный раздел отформатирован, Setup копирует на данный логический диск файлы Windows 2000, в том числе два стартовых файла, Ntldr и Ntdetect.com.

Еще одна задача программы установки — создание файла загрузочного меню, Boot.ini, в корневом каталоге загрузочного раздела. В этом файле содержатся параметры запуска устанавливаемой версии Windows 2000, а также сведения о всех системах, установленных до Windows 2000. Если файл Bootsect.dos содержит загрузочный сектор MS-DOS, в Boot.ini добавляется запись, позволяющая загружать MS-DOS. Ниже приведен пример файла Boot.ini с поддержкой двухвариантной загрузки для компьютера, на котором перед установкой Windows 2000 была установлена MS-DOS.

```
[boot loader]
timeout=30
default=multi(0)disk(0)rdisk(0)partition(1)\WINNT
[operating systems]
multi(0)disk(0)rdisk(0)partition(1)
\WINNT="Microsoft Windows 2000 Professional " /fastdetect
C:\="Microsoft Windows"
```

Загрузочный сектор и Ntldr

Перед тем как произвести запись в загрузочный сектор, программа установки должна выяснить формат раздела, поскольку от него зависит содержимое загрузочного сектора. Если это раздел FAT, Windows 2000 записывает в загрузочный сектор код, поддерживающий файловую систему FAT. Если раздел отформатирован под NTFS, в загрузочный сектор записывается код, соответствующий NTFS. Задача кода загрузочного сектора — предоставлять Windows 2000 информацию о структуре и формате логического диска и считывать из его корневого каталога файл Ntldr. После считывания Ntldr в память код загрузочного сектора передает управление в точку входа Ntldr. Если код загрузочного сектора не может найти Ntldr в корневом каталоге логического диска, он выводит сообщение об ошибке: «BOOT: Couldn't find NTLDR» (в FAT) или «NTLDR is missing» (в NTFS).

Ntldr начинает свою работу, когда система функционирует в реальном режиме (real mode) процессора *x*86. В реальном режиме трансляция между виртуальными и физическими адресами не осуществляется, поэтому программы, использующие какие-либо адреса памяти, интерпретируют их как физические. В этом режиме доступен лишь первый мегабайт физической памяти компьютера; в нем выполняются простые программы MS-DOS. Однако первое, что делает Ntldr, — переключает систему в защищенный режим (protected mode). На этой стадии трансляция между виртуальными адресами и физическими попрежнему отсутствует, но становится доступным полный объем памяти. Переключив систему в защищенный режим, Ntldr может работать со всей физической памятью. После того как он создает таблицы страниц, число которых достаточно для доступа к нижним 16 Мб памяти с подкачкой, Ntldr включает поддержку подкачки страниц. Защищенный режим с подкачкой страниц является нормальным режимом работы Windows 2000.

С этого момента Ntldr может работать в полнофункциональном режиме. Но при доступе к IDE-дискам и дисплею Ntldr все еще зависит от функций загрузочного кода, которые по мере необходимости на непродолжительное время отключают подкачку страниц и возвращают процессор в режим, позволяющий выполнять сервисы BIOS. Если загрузочные или системные диски являются SCSI-устройствами, Ntldr загружает файл Ntbootdd.sys и использует его функции обращения к диску вместо аналогичных функций загрузочного кода. Затем Ntldr с помощью встроенного кода файловой системы считывает из корневого каталога файл Boot.ini. В отличие от кода загрузочного сектора код Ntldr способен читать и подкаталоги.

Далее Ntldr очищает экран и, если в файле Boot.ini имеется более одной записи о доступных для загрузки операционных системах, выводит загрузочное меню. (Если в файле Boot.ini только одна запись, Ntldr пропускает загрузочное меню и сразу выводит стартовый индикатор прогресса загрузки.) Информация из Boot.ini адресует Ntldr к разделу, в котором находится системный каталог Windows 2000 (обычно \Winnt). Этим разделом может быть как загрузочный, так и другой главный раздел.

Если запись Boot.ini ссылается на MS-DOS, Ntldr считывает в память содержимое файла Bootsect.dos, переключается обратно в 16-разрядный реальный режим и вызывает из Bootsect.dos код MBR. В результате код из Bootsect.dos выполняется аналогично коду, считанному MBR с диска. Код из Bootsect.dos инициирует процесс загрузки, специфичный для MS-DOS. Так же происходит загрузка Microsoft Windows 98 или Microsoft Windows 95, если они установлены вместе с Windows 2000.

Записи Boot.ini могут включать ряд необязательных параметров, интерпретируемых Ntldr и другими компонентами в процессе загрузки. Полный список этих параметров приводится в таблице 4-2.

Таблица 4-2. *Параметры в Boot.ini*

Параметр	Описание
/3GB	Увеличивает пользовательскую часть адресного пространства процессов с 2 до 3 Гб, тем самым уменьшая размер системной части пространства с 2 до 1 Гб). Позволяет повысить производительность приложений, интенсивно использующих виртуальную память (например, серверов баз данных), предоставляя им большее адресное пространство. Но здесь нужно соблюсти еще два условия: система должна работать под управлением Windows 2000 Advanced Server или Datacenter Server, а в исполняемом файле приложения должен присутствовать флаг, указывающий на поддержку 3-гигабайтного пользовательского адресного пространства (см. раздел «Структура адресного пространства» главы 7).
/BASEVIDEO	Заставляет Windows 2000 использовать стандартный драйвер VGA-видеоадаптера для работы в GUI-режиме.
/BAUDRATE=	Включает отладку в режиме ядра и позволяет изменить устанавливаемую по умолчанию скорость передачи по соединению с удаленным хостом, используемым для отладки ядра (19200). Например: /BAUDRATE=115200.

см. след. стр.

Таблица 4-2. *продолжение*

Параметр	Описание
/BOOTLOG	Заставляет Windows 2000 вести журнал загрузки и записывать его в файл %SystemRoot%\Ntbtlog.txt.
/BREAK	Вызывает остановку HAL на точке прерывания в процессе инициализации. Первое, что делает ядро Windows 2000, — инициализирует HAL. Так что данная точка прерывания является первой из возможных. HAL может неопределенно долго ждать соединения с удаленным отладчиком ядра. При использовании этого параметра без /DEBUG появляется «синий экран» со стоп-кодом 0x00000078 (PHASE0_EXCEPTION).
/BURNMEMORY=	Указывает объем памяти, запрещаемой для использования операционной системой Windows 2000 (по аналогии с параметром /MAXMEM). Значение задается в мегабайтах. Например, /BURNMEMORY=128 означает, что 128 Мб физической памяти компьютера недоступны Windows 2000.
/CLKLVL	Перенастраивает стандартную многопроцессорную версию HAL для систем типа *x*86 (Halmps.dll) на распознавание сигналов системного таймера по потенциалу, а не по фронту.
/CRASHDEBUG	При загрузке системы загружает и отладчик ядра, который остается неактивным до момента краха. Этот параметр освобождает последовательный порт, который иначе был бы постоянно задействован отладчиком. Пока не произошел крах, порт может использоваться системой (параметр /DEBUG, напротив, заставляет отладчик ядра постоянно занимать последовательный порт).
/DEBUG	Включает отладку в режиме ядра.
/DEBUGPORT=	Включает отладку в режиме ядра и назначает порт для подключения удаленного хоста с отладчиком ядра, отличный от заданного по умолчанию (COM1). Например: /DEBUGPORT=COM2.
/FASTDETECT	Параметр загрузки Windows 2000 по умолчанию. Заменяет /NOSERIALMICE, применяемый в Windows NT 4. Введен для поддержки модулем NTDETECT альтернативной загрузки Windows NT 4. В Windows 2000 устройства, подключенные к параллельным и последовательным портам, определяются PnP-драйверами устройств, тогда как в Windows NT 4 эти функции возлагаются на NTDETECT. Параметр /FASTDETECT заставляет NTDETECT пропускать перечисление устройств, подключенных к параллельным и последовательным портам (не требуемое при загрузке Windows 2000). А в отсутствие этого параметра NTDETECT перечисляет такие устройства, что необходимо для загрузки Windows NT 4.
/INTAFFINITY	Указывает стандартной многопроцессорной версии HAL для систем типа *x*86 (Halmps.dll) так настроить привязку прерываний, чтобы лишь один процессор (с наибольшим порядковым номером) принимал запросы на прерывания. В отсутствие этого параметра (по умолчанию) HAL разрешает принимать запросы на прерывания всем процессорам.

Таблица 4-2. *продолжение*

Параметр	Описание
/KERNEL= /HAL=	Позволяет задавать имена файлов образа ядра и/или HAL, отличные от используемых по умолчанию (Ntoskrnl.exe и Hal.dll). Эти параметры предназначены для переключения между проверочными (отладочными) и рабочими версиями ядра, а также для выбора HAL вручную. Если Вы хотите загрузить отладочную среду, состоящую только из проверочных версий ядра и HAL (этого, как правило, достаточно для тестирования драйверов), выполните следующие операции. 1. Скопируйте отладочные версии ядра с дистрибутивного компакт-диска в каталог \Winnt\System32, изменив при этом их имена. Например, на однопроцессорной системе скопируйте Ntoskrnl.exe в Ntoschk.exe и Ntkrnlpa.exe в Ntoschkpa.exe, а на многопроцессорной — Ntkrnlmp.exe в Ntoschk.exe и Ntkrpamp.exe в Ntoschkpa.exe. Файлу ядра нужно присвоить краткое имя в формате «8.3». 2. Скопируйте с дистрибутивного компакт-диска \I386\Driver.cab в файл Halchk.dll в каталоге \Winnt\System32. Чтобы узнать, какую именно версию HAL нужно скопировать, найдите в \Winnt\Repair\Setup.log строку с Hal.dll. Она выглядит примерно так: *\Winnt\system32\hal.dll=«halacpi.dll», «1d8a1»*. Искомое имя файла HAL находится сразу после знака равенства. Файлу HAL нужно присвоить краткое имя в формате «8.3». 3. Сделайте копию строк, присутствовавших в файле Boot.ini по умолчанию. 4. В строку с определением элементов загрузочного меню добавьте новый элемент для отладочной среды (например, «Windows 2000 Professional Checked»). 5. В конец нового элемента добавьте /KERNEL=NTOSCHK.EXE /HAL= HALCHK.DLL. Учтите, что теперь в загрузочном меню доступны два элемента: новый — для загрузки отладочной среды и старый — для загрузки рабочей среды.
/MAXMEM=	Ограничивает объем памяти, доступный Windows 2000. Физическая память, лежащая за пределами указанного значения, системой игнорируется (не используется). Значение задается в мегабайтах. Например, при /MAXMEM=32 система использует только первые 32 Мб физической памяти.
/MAXPROCSPERCLUSTER=	При использовании стандартной многопроцессорной версии HAL для систем *x*86 (Halmps.dll) включает кластерный режим адресации контроллера прерываний APIC. Не поддерживается при использовании внешнего APIC-контроллера 82489DX.
/NODEBUG	Запрещает инициализацию отладки режима ядра. Имеет больший приоритет, чем остальные параметры, применяемые для отладки (/DEBUG, /DEBUGPORT и /BAUDRATE).

см. след. стр.

Таблица 4-2. *продолжение*

Параметр	Описание
/NOGUIBOOT	Запрещает инициализацию VGA-драйвера Windows 2000, ответственного за вывод растровой графики в процессе загрузки. Этот драйвер используется Windows 2000 для вывода информации о ходе загрузки, поэтому при его отключении Windows 2000 не будет выводить эту информацию.
/NOLOWMEM	Применяется только с параметром /PAE и только при наличии в системе более 4 Гб физической памяти. Если эти условия соблюдены, PAE-версия ядра Windows 2000, Ntkrnlpa.exe, не использует первые 4 Гб физической памяти. Для загрузки драйверов и приложений, а также для создания всех пулов памяти используется область выше этой границы. Данный параметр предназначен в основном для тестирования драйверов на совместимость с системами, в которых установлено более 4 Гб памяти.
/NOPAE	Заставляет Ntldr загружать версию ядра, не поддерживающую PAE (Physical Address Extensions) даже в том случае, когда система поддерживает такой механизм и имеет более 4 Гб физической памяти.
/NOSERIALMICE=[COM*x* | COM*x*,*y*,*z*...]	Устаревший спецификатор Windows NT 4, замененный на /FASTDETECT. Отключает детекцию мыши на указанных COM-портах. Применяется в том случае, если к последовательному порту подключено устройство, отличное от мыши. При использовании /NOSERIALMICE без указания порта детекция мыши отключается для всех COM-портов. См. также статью Q131976 в Microsoft Knowledge Base.
/NUMPROC=	Указывает число процессоров, которые Windows 2000 может использовать в многопроцессорной системе. Например, указав /NUMPROC=2 в четырехпроцессорной системе, Вы запретите использование двух из четырех процессоров.
/ONECPU	В многопроцессорных системах заставляет Windows 2000 работать только с одним процессором.
/PAE	Указывает Ntldr загрузить Ntkrnlpa.exe, версию ядра (для *x*86-процессоров), использующую преимущества PAE. PAE обеспечивает 64-разрядную адресацию драйверами устройств. Этот параметр предназначен для тестирования драйверов на совместимость с системами с большим объемом памяти.
/PCILOCK	Запрещает Windows 2000 динамически назначать PCI-устройствам IRQ и другие ресурсы для ввода-вывода. При этом используются настройки, заданные в BIOS. Подробности см. в статье Q148501 в Microsoft Knowledge Base.
/SAFEBOOT:	Задает параметры загрузки в безопасном режиме. Никогда не указывайте этот параметр вручную, так как Ntldr сам задает его при использовании меню, открываемого нажатием клавиши F8. (При загрузке Windows 2000 в безопасном режиме загружаются только драйверы и сервисы, указанные в подразделах реестра Minimal или Network, которые находятся в HKLM\SYSTEM\CurrentControlSet\Control\SafeBoot.) После двоеточия можно

Таблица 4-2. *продолжение*

Параметр	Описание
	указать один из трех дополнительных параметров: MINIMAL, NETWORK или DSREPAIR. Параметр MINIMAL соответствует безопасной загрузке без сетевой поддержки, NETWORK — безопасной загрузке с сетевой поддержкой, а DSREPAIR (Directory Services Repair) предназначен для перехода Windows 2000 в режим восстановления Active Directory с предоставляемой Вами резервной копии. Можно использовать еще один дополнительный параметр, ALTERNATESHELL — тогда вместо оболочки по умолчанию (Windows Explorer) загружается оболочка, указанная в разделе реестра HKLM\System\CurrentControlSet\SafeBoot\AlternateShell.
/SCSIORDINAL:	Указывает идентификатор SCSI-контроллера для Windows 2000 явным образом. (Добавление нового SCSI-устройства в систему с интегрированным SCSI-контроллером может привести к смене идентификатора этого контроллера.) Подробности см. в статье Q103625 в Microsoft Knowledge Base.
/SOS	Заставляет Windows 2000 перечислять на экране список помеченных к загрузке драйверов устройств, а затем показывать номер версии системы (включая номер сборки), объем физической памяти и число процессоров.
/TIMERES=	В системах со стандартной многопроцессорной версией HAL (Halmps.dll) задает разрешение системного таймера. Аргументом является значение в сотнях наносекунд, но частота устанавливается в соответствии с ближайшим меньшим значением, поддерживаемым HAL (см. ниже).

Сотни наносекунд	*Миллисекунды (мс)*
9766	0.98
19532	2.00
39063	3.90
78125	7.80

	Разрешение по умолчанию — 7,8 мс. Разрешение системного таймера влияет на разрешение ожидаемых таймеров. Например, параметр /TIMERES=21000 установит разрешение таймера равным 2,0 мс.
/USE8254	Указывает HAL использовать в качестве базового таймера чип 8254. Подробности см. в статье Q169901 в Microsoft Knowledge Base.
/WIN95	Указывает Ntldr загрузить содержимое файла Bootsect.w40, в котором хранится загрузочный сектор Windows 9*x*. Применим, только если на компьютере установлены три операционные системы (Windows 2000, Windows 9*x* и MS-DOS). Подробности см. в статье Q157992 в Microsoft Knowledge Base.
/WIN95DOS	Указывает Ntldr использовать загрузочный сектор MS-DOS, хранящийся в файле Bootsect.dos. Применим, только если на компьютере установлены три операционные системы (Windows 2000, Windows 9*x* и MS-DOS). Подробности см. в статье Q157992 в Microsoft Knowledge Base.

см. след. стр.

Таблица 4-2. *продолжение*

Параметр	Описание
/YEAR=	Указывает Windows 2000 игнорировать значение года, сообщаемое системными часами компьютера и использовать заданное значение. Этот параметр влияет на все программное обеспечение, в том числе на ядро. (Например, /YEAR=2001.) Этот параметр предназначался для тестирования на совместимость с датами 2000 года.

Если до истечения периода ожидания, указанного в Boot.ini, пользователь не выбрал ни одной команды загрузочного меню, Ntldr выбирает вариант по умолчанию. После выбора одного из вариантов Ntldr загружает и запускает Ntdetect.com, 16-разрядную программу реального режима, которая получает от BIOS сведения о базовых устройствах и конфигурации компьютера:

- время и дату, хранящиеся в энергонезависимой памяти CMOS;
- типы шин в системе (например, ISA, PCI, EISA, MCA) и устройствах, подключенных к этим шинам;
- число, емкость и тип дисков, присутствующих в системе;
- тип подключенной мыши;
- число и тип параллельных портов, сконфигурированных в системе.

Эти сведения, записываемые во внутренние структуры данных, на более поздних этапах загрузки будут сохранены в разделе реестра HKLM\HARDWARE\DESCRIPTION.

Затем Ntldr очищает экран и выводит индикатор прогресса загрузки с надписью «Starting Windows» («Запуск Windows»). Индикатор остается на нулевой отметке до начала загрузки драйверов устройств (п. 5 в следующем списке). Под индикатором появляется сообщение: «For troubleshooting and advanced startup options for Windows 2000, press F8» («Для выбора особых вариантов загрузки Windows 2000 нажмите F8»). При нажатии клавиши F8 выводится дополнительное загрузочное меню, предлагающее выбрать особые варианты загрузки — последнюю удачную конфигурацию, безопасный или отладочный режим и т. д.

Далее Ntldr начинает загружать необходимые для инициализации ядра файлы.

1. Загружает соответствующие образы ядра и HAL (по умолчанию — Ntoskrnnl.exe и Hal.dll). Если Ntldr не удается загрузить какой-либо из этих файлов, он выводит сообщение «Windows 2000 could not start because the following file was missing or corrupt» («Не удается запустить Windows 2000 из-за испорченного или отсутствующего файла»), за которым следует имя файла.
2. Для поиска драйверов устройств, которые нужно загрузить, считывает в память содержимое куста реестра SYSTEM, \Winnt\System32\Config\System. Куст (hive) — это файл, содержащий поддерево реестра. Подробнее о реестре см. главу 5.
3. Сканирует загруженный в память куст реестра SYSTEM и находит все загрузочные драйверы устройств (это драйверы, обязательные для запуска

системы). Они отмечены в реестре флагом SERVICE_BOOT_START. Каждому драйверу устройства в реестре соответствует подраздел HKLM\SYSTEM\CurrentControlSet\Services. Например, драйверу диспетчера логических дисков (Logical Disk Manager) в разделе Services соответствует подраздел Dmio, показанный на рис. 4-2. (Подробное описание содержимого Services см. в разделе «Сервисы» главы 5.)

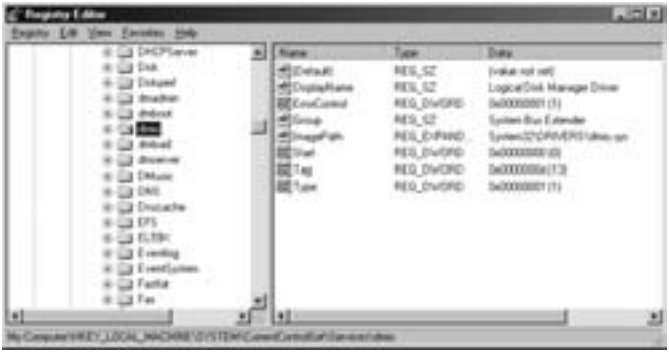

Рис. 4-2. *Параметры драйвера диспетчера логических дисков*

4. Вносит в список загрузочных драйверов устройств драйвер файловой системы, отвечающий за реализацию кода для конкретного типа раздела (FAT, FAT32 или NTFS), на котором находится системный каталог. Ntldr должен загрузить этот драйвер именно сейчас, иначе ядро будет требовать от драйверов их же загрузки, и получится замкнутый круг.
5. Загружает драйверы, обязательные для запуска системы. Ход загрузки отражается индикатором «Starting Windows». Полоска на индикаторе продвигается вперед по мере загрузки драйверов (число загрузочных драйверов считается равным 80, поэтому после успешной загрузки каждого драйвера полоска продвигается на 1,25%). Если в Boot.ini указан параметр /SOS, то вместо индикатора Ntldr выводит имя файла каждого загрузочного драйвера. Учтите, что 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.

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), с указателем на загруженные в память копии кустов реестра HARDWARE и SYSTEM и с указателем 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 *KiSystemStartup*, которая в свою очередь вызывает *HalInitializeProcessor* и *KiInitializeKernel* для каждого процессора. Работая на стартовом процессоре, *KiInitializeKernel* выполняет общесистемную инициализацию ядра, в том числе всех внутренних структур данных, разделяемых всеми процессорами. Затем каждый экземпляр *KiInitializeKernel* вызывает функцию *ExpInitializeExecutive*, отвечающую за управление фазой 0.

ExpInitializeExecutive начинает с вызова HAL-функции *HalInitSystem*, позволяющей HAL взять управление инициализацией системы на себя. Одной из задач *HalInitSystem* является подготовка системного контроллера прерываний каждого процессора к обработке прерываний и конфигурирование таймера, используемого для учета распределяемого процессорного времени (подробнее на эту тему см. раздел «Учет квантов времени» главы 6).

Лишь на стартовом процессоре *ExpInitializeExecutive* не просто вызывает *HalInitSystem*, но и выполняет другие операции по инициализации. Когда *HalInitSystem* возвращает управление, функция *ExpInitializeExecutive*, выполняемая на стартовом процессоре, обрабатывает параметр /BURNMEMORY файла Boot.ini (если таковой указан и действителен для данного варианта загрузки). В соответствии с этим параметром *ExpInitializeExecutive* исключает указанный объем памяти.

Далее *ExpInitializeExecutive* вызывает процедуры инициализации для диспетчера памяти, диспетчера объектов, справочного монитора безопасности, диспетчера процессов и диспетчера Plug and Play. 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. Другие компоненты исполнительной системы, ядро и драйверы устройств пользуются этими пулами, выделяя память под собственные структуры данных.
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 описателей для поддержки учета ресурсов.
3. Справочный монитор безопасности инициализирует объект типа «маркер доступа» и использует его для создания и подготовки первых маркеров, назначаемых начальным процессам.
4. Диспетчер процессов производит бóльшую 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 *Phase1Initialization*. Этот поток не запускается сразу же после создания, поскольку прерывания пока запрещены.

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

Когда на каждом процессоре управление возвращается к функции *KiInitializeKernel*, она передает его циклу Idle. В результате системный поток, созданный, как было описано в п. 4 предыдущего списка, начинает фазу 1. Дополнительные процессоры ждут начала своей инициализации до п. 5 фазы 1 (см. список ниже). Вот какие операции выполняются в фазе 1.

1. Для подготовки системы к приему прерываний от устройств и для разрешения прерываний вызывается *HalInitSystem*.
2. Вызывается загрузочный видеодрайвер (\Winnt\System32\Bootvid.dll), который выводит экран-заставку, показываемую в процессе запуска Windows 2000.
3. Инициализируется диспетчер электропитания.
4. Инициализируются системные часы (вызовом *HalQueryRealTimeClock*), текущее значение которых сохраняется как время загрузки системы.
5. В многопроцессорной системе инициализируются остальные процессоры и начинается выполнение команд.
6. Индикатор прогресса загрузки устанавливается в отметку 5%.
7. Диспетчер объектов создает корневой каталог пространства имен и каталоги \ObjectTypes, \??, а также ссылку \DosDevice на \??.
8. Вызывается исполнительная система для создания своих типов объектов, включая семафор, мьютекс, событие и таймер.
9. Ядро инициализирует структуры данных планировщика (диспетчера) и таблицу диспетчеризации системных сервисов.
10. Справочный монитор безопасности создает в пространстве имен диспетчера объектов каталог \Security и инициализирует структуры данных аудита (если аудит системы разрешен).
11. Индикатор прогресса загрузки устанавливается в отметку 10%.
12. Для создания объекта «раздел» и системных рабочих потоков вызывается диспетчер памяти (см. главу 7).
13. На системное адресное пространство проецируются таблицы NLS (National Language Support).
14. На системное адресное пространство проецируется Ntdll.dll.
15. Диспетчер кэша инициализирует структуры данных кэша файловой системы и создает свои рабочие потоки.
16. Диспетчер конфигурации создает в пространстве имен объект «раздел реестра» \Registry и копирует переданные Ntldr начальные данные в кусты реестра HARDWARE и SYSTEM.
17. Инициализируются структуры данных драйвера файловой системы.
18. Диспетчер Plug and Play вызывает PnP BIOS.
19. Индикатор прогресса загрузки устанавливается в отметку 20%.
20. Подсистема LPC инициализирует объект типа «порт LPC».

21. Если система запущена с протоколированием загрузки (/BOOTLOG), инициализируется файл протокола загрузки.
22. Индикатор прогресса загрузки устанавливается в отметку 25%.
23. Наступает момент инициализации диспетчера ввода-вывода. Согласно показаниям индикатора, эта стадия запуска системы занимает 50% времени. После успешной загрузки каждого драйвера диспетчер ввода-вывода продвигает полоску на индикаторе на 2% (если загружается более 25 драйверов, индикатор останавливается на отметке 75%).

    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. (Подробнее этот сложный процесс, специфичный для конкретной подсистемы ввода-вывода, рассматривается в главе 9.) Далее инициализируется подсистема WMI (Windows Management Instrumentation), которая предоставляет WMI-поддержку WDM-драйверам устройств. (Подробнее о WMI см. раздел «Windows Management Instrumentation» главы 5.) После этого вызываются все загрузочные драйверы, которые осуществляют свою инициализацию. Также загружаются и инициализируются драйверы, необходимые для запуска системы (см. главу 9). Наконец, в пространстве имен диспетчера объектов создаются имена устройств MS-DOS в виде символьных ссылок.
24. Индикатор прогресса загрузки устанавливается в отметку 75%.
25. Если система загружается в безопасном режиме, этот факт отмечается в реестре.
26. Включается подкачка страниц для кода режима ядра (в Ntkrnl и драйверах), если она явно не запрещена в реестре.
27. Индикатор прогресса загрузки устанавливается в отметку 80%.
28. Вызывается диспетчер электропитания для инициализации своих структур данных.
29. Индикатор прогресса загрузки устанавливается в отметку 85%.
30. Вызывается справочный монитор безопасности для создания потока Command Server, взаимодействующего с Lsass (см. раздел «Компоненты системы защиты» главы 8).
31. Индикатор прогресса загрузки устанавливается в отметку 90%.
32. На завершающем этапе создается процесс Smss диспетчера сеансов (см. главу 2). Smss отвечает за создание среды пользовательского режима, которая предоставляет визуальный интерфейс Windows 2000. Об инициализации Smss см. следующий раздел.
33. Индикатор прогресса загрузки устанавливается в отметку 100%.

Перед завершением инициализации компонентов исполнительной системы и ядра поток инициализации фазы 1 в течение пяти секунд ждет освобождения описателя процесса Smss. Если этот процесс завершается до истечения пяти секунд, происходит крах системы с кодом SESSION5_INITIALIZATION_FAILED.

По истечении пяти секунд запуск диспетчера сеансов считается успешным, и вызывается функция потока обнуления страниц диспетчера памяти (см. главу 7).

Smss, Csrss и Winlogon

Smss похож на любой другой процесс пользовательского режима, но имеет два существенных отличия. Во-первых, Windows 2000 считает его *доверяемой* (trusted) 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 Win32 API, а базовые API-функции исполнительной системы, в совокупности называемые Windows 2000 Native API. Smss не обращается к Win32 API, поскольку при его запуске подсистема Win32 еще не функционирует. Запуск подсистемы Win32 и является одной из его первых задач.

Затем Smss вызывает диспетчер конфигурации, который завершает инициализацию реестра, заполняя все его разделы. Диспетчер конфигурации запрограммирован так, что ему известно местонахождение всех кустов реестра на диске, кроме содержащих пользовательские параметры. Список всех загружаемых им кустов реестра хранится в разделе HKLM\SYSTEM\CurrentControlSet\Control\Hivelist.

Основной поток Smss выполняет следующие инициализирующие операции.

1. Создает объект «порт LPC» (*SmApiPort*) и два потока, ожидающие клиентские запросы (например, на загрузку новой подсистемы или на создание сеанса).
2. Определяет символьные ссылки на имена устройств MS-DOS (вроде COM1 и LPT1).
3. Если установлены Terminal Services, создает в пространстве имен диспетчера объектов каталог \Sessions (для нескольких сеансов).
4. Запускает программы, указанные в HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\BootExecute. Как правило, в нем содержится одна команда на запуск Autochk (версия Chkdsk, работающая на этапе загрузки).
5. Выполняет отложенные действия по переименованию файлов, указанные в разделе HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\PendingFileRenameOperations. Отложенные операции по удалению файлов указываются в подразделе PendingFileRenameOperations2.
6. Открывает известные DLL.
7. Создает дополнительные страничные файлы.
8. Инициализирует реестр. Диспетчер конфигурации заполняет реестр, загружая кусты HKLM\SAM, HKLM\SECURITY и HKLM\SOFTWARE. Хотя информация о местонахождении файлов кустов содержится в разделе HKLM\SYSTEM\CurrentControlSet\Control\Hivelist, диспетчер конфигурации ищет эти файлы в каталоге \Winnt\System32\Config.
9. Создает системные переменные окружения.
10. Загружает часть подсистемы Win32, работающую в режиме ядра (Win32k.sys). Smss определяет местонахождение Win32k.sys и других загружаемых им

компонентов по путям, хранящимся в HKLM\SYSTEM\CurrentControlSet\Control\Session Manager. 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.

11. Запускает процессы подсистем, в том числе Csrss. (Как говорилось в главе 2, подсистемы POSIX и OS/2 запускаются по требованию.)
12. Запускает процесс Winlogon. Этапы запуска Winlogon кратко описываются ниже.
13. Создает порты LPC для отладочных сообщений (*DbgSsApiPort* и *DbgUiApiPort*) и потоки, прослушивающие эти порты.

После выполнения вышеперечисленных операций основной поток Smss переходит к бесконечному ожиданию описателей процессов Csrss и Winlogon. Поскольку от этих процессов зависит функционирование Windows 2000, в случае их неожиданного завершения Smss вызывает крах системы.

Далее Winlogon продолжает инициализацию, выполняя следующие операции: создает начальный объект WindowStation и объекты рабочего стола, загружает GINA DLL и т. д. Затем он создает процесс SCM (диспетчера управления сервисами) (\Winnt\System32\Services.exe), который загружает все сервисы и драйверы устройств, помеченные для автоматического запуска, а также запускает процесс Lsass (подсистемы локальной аутентификации) (\Winnt\System32\Lsass.exe). Подробнее о запуске Winlogon и Lsass см. раздел «Инициализация Winlogon» главы 8.

После того как SCM инициализирует автоматически запускаемые сервисы и драйверы устройств, а пользователь успешно зарегистрируется в системе, загрузка считается успешно завершенной. Параметры в разделе HKLM\SYSTEM\Select\LastKnownGood обновляются в соответствии со значениями параметров последней удачной конфигурации (\CurrentControlSet). Если пользователь при следующем запуске выберет в загрузочном меню вариант загрузки последней удачной конфигурации или если какой-нибудь драйвер приведет к критической ошибке, система использует текущие параметры из подраздела \LastKnownGood. Поскольку с этими параметрами система успешно загружалась минимум один раз, их применение в случае сбоя значительно повышает вероятность успешного запуска.

И на этом процесс загрузки заканчивается.

Безопасный режим

Наиболее распространенная причина, по которой системы с Windows 2000 становятся незагружаемыми, заключается в том, что какой-то драйвер устройства приводит к краху при загрузке. Поскольку со временем программно-аппаратная конфигурация системы может измениться, скрытые до этого ошибки в драйверах могут проявиться в любой момент. Windows 2000 предоставляет администратору способ решения подобных проблем: загрузку в *безопасном режиме* (safe mode). Понятие безопасного режима в Windows 2000 заимствовано из потребительских версий Windows и представляет собой конфигура-

цию с минимальным набором драйверов и сервисов. Используя только самые необходимые драйверы, Windows 2000 избегает загрузки сторонних драйверов, способных вызывать крах системы.

Нажав клавишу F8 в начале загрузки Windows 2000, Вы открываете дополнительное загрузочное меню, в котором присутствуют три варианта загрузки в безопасном режиме: Safe Mode (Безопасный режим), Safe Mode With Networking (Безопасный режим с загрузкой сетевых драйверов) и Safe Mode With Command Prompt (Безопасный режим с поддержкой командной строки). Стандартный безопасный режим подразумевает использование минимума необходимых для успешной загрузки драйверов. В безопасном режиме с сетевой поддержкой дополнительно загружаются сетевые драйверы и сервисы. Наконец, единственное отличие безопасного режима с поддержкой командной строки от стандартного заключается в том, что Windows 2000 вместо обычной оболочки Windows Explorer, позволяющей работать в GUI-режиме, запускает оболочку командной строки (Cmd.exe).

В Windows 2000 предусмотрен и четвертый безопасный режим — Directory Services Restore (Восстановление службы каталогов), который применяется для восстановления службы каталогов Active Directory на контроллере домена с резервной копии. Поскольку в этом режиме загружается весь набор драйверов и сервисов, его нельзя использовать для запуска системы, которая перестала загружаться в нормальном режиме.

Загрузка драйверов в безопасном режиме

Как Windows 2000 определяет набор драйверов для загрузки в стандартном безопасном режиме и безопасном режиме с сетевой поддержкой? Ответ следует искать в содержимом раздела реестра HKLM\SYSTEM\CurrentControlSet\Control\SafeBoot. В нем присутствуют подразделы Minimal и Network. Каждый подраздел в свою очередь содержит подразделы с именами драйверов, сервисов или их групп. Так, подраздел vga.sys определяет драйвер видеоадаптера VGA, который поддерживает базовый набор графических сервисов стандартного видеоадаптера для IBM-совместимого компьютера. Этот драйвер используется системой в безопасном режиме вместо драйверов, которые позволяют задействовать все преимущества куда более совершенных видеоадаптеров, но способны помешать успешной загрузке системы. Каждому подразделу в разделе SafeBoot соответствует параметр по умолчанию, описывающий, что именно идентифицирует данный подраздел. Например, в подразделе vga.sys параметр по умолчанию — Driver.

Параметром по умолчанию для подраздела файловой системы является Driver Group. При создании сценария установки для драйвера устройства разработчик может указать, что он относится к какой-либо группе драйверов. Группы драйверов, определенные в системе, перечисляются в параметре List раздела реестра HKLM\SYSTEM\CurrentControlSet\Control\ServiceGroupOrder. Разработчик приписывает драйвер к той или иной группе, чтобы указать Windows 2000, на каком этапе загрузки следует запускать данный драйвер. Главное предназначение раздела ServiceGroupOrder — определение порядка загрузки групп драйверов. Некоторые группы драйверов нужно загружать до или после других. Параметр Group в подразделе реестра со сведениями о конфи-

гурации драйвера, сопоставляет его с определенной группой. Подразделы со сведениями о конфигурации драйверов и сервисов находятся в разделе HKLM\SYSTEM\CurrentControlSet\Services. Взглянув на его содержимое, Вы найдете раздел VgaSave для драйвера видеоадаптера VGA, который принадлежит к группе Video Save. Любой драйвер файловой системы, необходимый Windows 2000 для обращения к системному диску, находится в группе Boot File System. Если файловой системой такого диска является NTFS, то в группу входит драйвер NTFS; в ином случае в группу входит драйвер Fastfat (поддерживающий диски FAT12, FAT16 и FAT32). Другие драйверы файловой системы входят в группу File System, которая также включена в конфигурации Safe Mode и Safe Mode With Networking.

При загрузке в безопасном режиме Ntldr передает ядру (Ntoskrnl.exe) вместе с другими параметрами, указанными в Boot.ini для текущего варианта загрузки, параметр командной строки /SAFEBOOT:, добавляя к нему одну или несколько строк в зависимости от выбранного типа безопасного режима. Для стандартного безопасного режима Ntldr добавляет MINIMAL, для Safe Mode With Networking — NETWORK, для Safe Mode With Command Prompt — MINIMAL(ALTERNATESHELL), а для Directory Services Restore — DSREPAIR.

Ядро Windows 2000 сканирует параметры загрузки в поисках спецификаторов безопасного режима и устанавливает значение внутренней переменной *InitSafeBootMode* в соответствии с результатом поиска. Значение этой переменной также записывается в раздел HKLM\SYSTEM\CurrentControlSet\Control\SafeBoot\Option\OptionValue, что позволяет компонентам пользовательского режима (например, SCM), определять режим загрузки системы. Кроме того, при выборе Safe Mode With Command Prompt, ядро присваивает значение 1 параметру UseAlternateShell в разделе реестра HKLM\SYSTEM\CurrentControlSet\Control\SafeBoot\Option. Кроме того, ядро записывает параметры, переданные Ntldr, в раздел HKLM\SYSTEM\CurrentControlSet\Control\SystemStartOptions.

Когда диспетчер ввода-вывода загружает драйверы устройств, указанные в разделе HKLM\SYSTEM\CurrentControlSet\Services, он выполняет функцию *IopLoadDriver*. А когда диспетчер Plug and Play обнаруживает новое устройство и хочет динамически загрузить драйвер для этого устройства, он вызывает функцию *IopCallDriverAddDevice*. Обе эти функции перед загрузкой драйвера обращаются к функции *IopSafeBootDriverLoad*. Последняя проверяет значение переменной *InitSafeBootMode* и определяет, можно ли загрузить данный драйвер. Так, если система загружается в стандартном безопасном режиме, *IopSafeBootDriverLoad* ищет группу этого драйвера (если таковая есть) в подразделе Minimal. Найдя ее, *IopSafeBootDriverLoad* уведомляет вызвавшую функцию о том, что этот драйвер можно загрузить. В ином случае *IopSafeBootDriverLoad* ищет в том же подразделе имя драйвера. Если оно есть в списке, драйвер может быть загружен. Если *IopSafeBootDriverLoad* не находит в списке группу или имя данного драйвера, его загрузка запрещается. При загрузке системы в безопасном режиме с сетевой поддержкой *IopSafeBootDriverLoad* ведет поиск в подразделе Network, а в случае загрузки системы в нормальном режиме *IopSafeBootDriverLoad* разрешает загрузку всех драйверов.

Однако Ntldr загружает все драйверы, у которых в соответствующих разделах реестра значение Start равно 0, что указывает на необходимость их за-

грузки при запуске системы. Поскольку Ntldr не проверяет раздел SafeBoot, он загружает все загрузочные драйверы.

Программное обеспечение с поддержкой безопасного режима

SCM (Services.exe), проводя инициализацию при загрузке, проверяет параметр OptionValue в разделе реестра HKLM\SYSTEM\CurrentControlSet\Control\SafeBoot\Option, чтобы выяснить, загружается ли система в безопасном режиме. Если да, SCM зеркально воспроизводит действия *IopSafeBootDriverLoad*. Он обрабатывает все сервисы, перечисленные в HKLM\SYSTEM\CurrentControlSet\Services, но загружает лишь отмеченные в соответствующем подразделе реестра для загрузки в безопасном режиме. Подробнее об инициализации SCM см. раздел «Сервисы» главы 5.

Userinit (\Winnt\System32\Userinit.exe) — другой компонент пользовательского режима, которому нужно знать, загружается ли система в безопасном режиме. Userinit, инициализирующий среду для пользователя при его входе в систему, проверяет значение HKLM\SYSTEM\CurrentControlSet\Control\SafeBoot\UseAlternateValue. Если это значение установлено, в качестве пользовательской оболочки он запускает не Explorer.exe, а программу, указанную в HKLM\SYSTEM\CurrentControlSet\Control\SafeBoot\AlternateShell. Когда Вы устанавливаете Windows 2000 на компьютер, параметру AlternateShell присваивается значение Cmd.exe, и командная строка Win32 становится оболочкой по умолчанию для безопасного режима с командной строкой. Но, даже если текущей оболочкой является командная строка, из нее можно запустить Windows Explorer, введя команду *Explorer.exe*. Аналогичным образом из командной строки можно запустить любую GUI-программу.

А как приложения узнают о загрузке системы в безопасном режиме? Вызовом Win32-функции *GetSystemMetrics(SM_CLEANBOOT)*. Пакетные сценарии, выполняющие некоторые действия при загрузке системы в безопасном режиме, проверяют наличие переменной окружения SAFEBOOT_OPTION, так как система определяет ее только при загрузке в безопасном режиме.

Ведение протокола при загрузке в безопасном режиме

Если Вы загружаете систему в безопасном режиме, Ntldr передает ядру Windows 2000 вместе с параметрами, устанавливающими безопасный режим, и параметр /BOOTLOG. При инициализации ядро проверяет наличие параметра /BOOTLOG независимо от того, задан ли безопасный режим. Если ядро обнаруживает соответствующую строку, оно протоколирует все свои действия при загрузке каждого драйвера. Так, если функция *IopSafeBootDriverLoad* запрещает загрузку какого-либо драйвера, диспетчер ввода-вывода вызывает функцию *IopBootLog*, регистрируя, что данный драйвер не загружен. Аналогичным образом после успешной загрузки драйвера, входящего в конфигурацию безопасного режима, *IopLoadDriver* вызывает *IopBootLog* для внесения записи о загрузке этого драйвера. Изучив файлы протокола загрузки, можно выяснить, какие драйверы являются частью данной конфигурации.

Поскольку ядро избегает изменения данных на диске до запуска Chkdsk, который происходит на более позднем этапе загрузки, *IopBootLog* не может просто сбрасывать записи в файл. Вместо этого она записывает их в раздел

реестра HKLM\SYSTEM\CurrentControlSet\BootLog. Диспетчер сеансов (Smss), первый загружаемый компонент пользовательского режима, запускает Chkdsk для проверки целостности системного диска, а потом завершает инициализацию реестра, вызывая *NtInitializeRegistry*. Этот вызов указывает ядру, что уже можно безопасно открыть на диске файл протокола, что и делается вызовом *IopCopyBootLogRegistryToFile*. Эта функция создает в системном каталоге Windows 2000 (по умолчанию — \Winnt) файл Ntbtlog.txt и копирует в него содержимое раздела реестра BootLog. *IopCopyBootLogRegistryToFile* также устанавливает флаг, сигнализирующий *IopBootLog* о возможности записи непосредственно в файл протокола. Ниже показан фрагмент образца такого файла.

```
Microsoft (R)Windows 2000 (R)Version 5.0 (Build 2195)
 2 11 2000 10:53:27.500
Loaded driver \WINNT \System32 \ntoskrnl.exe
Loaded driver \WINNT \System32 \hal.dll
Loaded driver \WINNT \System32 \BOOTVID.DLL
Loaded driver ACPI.sys
Loaded driver \WINNT \System32 \DRIVERS \WMILIB.SYS
Loaded driver pci.sys
Loaded driver isapnp.sys
Loaded driver compbatt.sys
Loaded driver \WINNT \System32 \DRIVERS \BATTC.SYS
Loaded driver intelide.sys
Loaded driver \WINNT \System32 \DRIVERS \PCIIDEX.SYS
Loaded driver pcmcia.sys
Loaded driver ftdisk.sys
Loaded driver Diskperf.sys
Loaded driver dmload.sys
Loaded driver dmio.sys
⋮
Did not load driver Media Control Devices
Did not load driver Communications Port
Did not load driver Audio Codecs
⋮
```

Консоль восстановления

В безопасном режиме обычно удается восстановить систему, ставшую незагружаемой в нормальном режиме из-за неправильной работы какого-либо драйвера устройства. Однако в некоторых ситуациях это не помогает: система все равно не загружается. Так, если сбойный драйвер входит в группу Safe, загрузиться в безопасном режиме не удастся. Другая ситуация, когда загрузка в безопасном режиме не помогает, — сбои в загрузочном драйвере стороннего поставщика, например в драйвере антивирусного сканера, поскольку загрузочные драйверы стартуют независимо от режима загрузки системы. Аналогичная ситуация возникает при повреждении файлов системных модулей или драйверов, входящих в конфигурацию безопасного режима, а также главной загрузочной записи (MBR) системного диска. Эти проблемы можно решить с помощью Recovery Console (Консоль восстановления). Консоль восстановления позволяет загрузить компактную оболочку командной строки с

дистрибутивного компакт-диска Windows 2000 (или ранее подготовленных загрузочных дискет) и восстановить систему без загрузки компьютера с жесткого диска.

При загрузке системы с дистрибутивного компакт-диска Windows 2000 появляется экран, на котором можно выбрать между установкой Windows 2000 и восстановлением существующей системы. При выборе второго варианта предлагается вставить дистрибутивный компакт-диск Windows 2000 (если он не вставлен в привод CD-ROM). Далее Вы должны выбрать один из двух вариантов восстановления: запустить консоль восстановления или начать процесс аварийного восстановления. Если при появлении экрана Setup Welcome (Вас приветствует программа установки) Вы нажмете клавишу F10, это меню выводиться не будет, а сразу запустится консоль восстановления.

После запуска консоль восстановления сканирует жесткие диски и формирует список систем Windows NT и Windows 2000 на данном компьютере (если они есть). Выбрав нужную систему, Вы должны ввести пароль, соответствующий учетной записи администратора для данной системы. Если регистрация прошла успешно, система предоставляет Вам командную оболочку, аналогичную среде MS-DOS. Гибкий набор команд позволяет выполнять простые операции с файлами (вроде копирования, переименования и удаления), включать и отключать службы и драйверы и даже восстанавливать MBR и загрузочные записи. Однако консоль восстановления обеспечивает доступ только к корневому каталогу, к каталогу, в котором установлена система и в котором Вы сейчас зарегистрировались, и к каталогам на сменных дисках, например на компакт-дисках или 3,5-дюймовых дискетах. Эти ограничения диктуются требованиями защиты данных, право на доступ к которым может отсутствовать у администратора одной из систем.

Для поддержки таких команд файлового ввода-вывода, как Cd, Rename или Move, консоль восстановления использует встроенный интерфейс системных вызовов Windows 2000. Команды Enable и Disable, позволяющие изменять режимы запуска драйверов устройств и сервисов (служб), работают иначе. Например, когда Вы командуете консоли восстановления отключить какой-либо драйвер, она обращается к разделу реестра Services и присваивает параметру Start в подразделе для соответствующего драйвера значение SERVICE_DISABLED. В результате при следующей загрузке системы данный драйвер загружаться не будет. Консоль также загружает куст реестра SYSTEM (\Winnt\System32\Config\System) для текущей системы, где в разделе HKLM\SYSTEM\CurrentControlSet\Services хранится нужная информация.

Когда Вы загружаете систему с дистрибутивного компакт-диска Windows 2000 или загрузочных дискет, к моменту появления экрана, предлагающего выбор между установкой или восстановлением Windows 2000, происходит загрузка с компакт-диска стартовой копии ядра Windows 2000 и всех драйверов поддержки (например, драйверов NTFS, FAT, SCSI и видеоадаптера). На компьютерах с процессорами типа *x*86 загрузка с компакт-диска управляется файлом Txtsetup.sif из каталога I386. В нем содержится список файлов, подлежащих загрузке, с указанием их местонахождения на компакт-диске. Как и при загрузке Windows 2000 с жесткого диска, первой запускаемой программой пользовательского режима является диспетчер сеансов (Smss.exe), распо-

ложенный в каталоге I386\System32. Диспетчер сеансов, используемый программой установки Windows 2000, отличается от стандартного в уже установленной системе. Эта версия диспетчера сеансов предоставляет меню, позволяющие установить или восстановить Windows 2000, а также выбрать тип восстановления. В процессе установки Windows 2000 этот компонент также помогает выбрать раздел для установки системы и копирует файлы на жесткий диск.

После запуска консоли восстановления диспетчер сеансов загружает и запускает два драйвера устройств, реализующие эту консоль: Spcmdcon.sys и Setupdd.sys. Первый предоставляет интерактивную командную строку и обрабатывает высокоуровневые команды. Второй является драйвером поддержки, предоставляющим Spcmdcon.sys набор функций для управления разделами диска, загрузки кустов реестра и управления видеовыводом. Setupdd.sys также взаимодействует с драйверами дисковых устройств для управления разделами и выводит на экран сообщения, используя базовую видеоподдержку, встроенную в ядро Windows 2000.

Консоль восстановления, приняв от Вас пароль для входа в выбранную систему, должна проверить его, даже несмотря на то что подсистема защиты Windows 2000 сейчас не функционирует. Таким образом, проверка пароля возлагается исключительно на консоль восстановления. Для этого консоль прежде всего загружает с жесткого диска (через Setupdd.sys) куст реестра диспетчера учетных записей безопасности (Security Accounts Manager, SAM) данной системы, где хранится информация о паролях. Куст SAM находится в каталоге \Winnt\System32\Config\Sam. После загрузки этого куста консоль восстановления находит в реестре системный ключ для расшифровки копии SAM в памяти. Шифрование куста SAM введено, начиная с Windows NT 4 Service Pack 3, для защиты от попыток взлома из MS-DOS.

Далее консоль восстановления (Spcmdcon.sys) отыскивает в SAM пароль для учетной записи Administrator (Администратор). На заключительном этапе проверки консоль хэширует пароль по алгоритму MD5, шифрует полученный хэш по алгоритму RC4 с применением системного ключа, а затем сравнивает полученный шифрованный хэш с зашифрованным хэшем из SAM. Если они совпадают, консоль восстановления считает, что Вы успешно вошли в систему, в ином случае консоль восстановления отказывает Вам в доступе.

Завершение работы системы

Если в системе зарегистрирован пользователь и какой-то процесс инициирует завершение работы системы, вызывая Win32-функцию *ExitWindowsEx*, Csrss получает сообщение о необходимости завершения системы. Тогда Csrss в интересах инициатора завершения системы посылает скрытому окну, которое принадлежит Winlogon, Windows-сообщение с требованием завершить работу системы. Winlogon, олицетворяющий зарегистрированного в данный момент пользователя (чей контекст защиты может совпадать, а может и не совпадать с контекстом защиты пользователя процесса, инициировавшего завершение работы), вызывает *ExitWindowsEx* с набором специальных внутренних флагов. В результате Csrss получает еще одно сообщение с запросом на завершение системы.