boot(1M)

НАЗВАНИЕ

boot - запуск ядра операционной системы или независимой программы

СИНТАКСИС

SPARC

```
boot [имена_OBP] [файл] [-afV] [-D стандартный_файл]
  [флаги_загрузки] [--] [аргументы_клиентской_программы]

b [устройство [(c, u, p)]] [-afV] [ -D стандартный_файл]
  [флаги_загрузки] [--] [аргументы_клиентской_программы]
```

IA

```
b [ файл ] [ -f ] [ аргументы_загрузки ]
i
```

ОПИСАНИЕ

Начальная загрузка (bootstrapping) - это процесс загрузки и выполнения независимой программы. В рамках обсуждаемой темы, начальная загрузка означает процесс загрузки и выполнения операционной системы. Обычно в качестве независимой программы выступает ядро операционной системы (см. **kernel(1M)**), но вместо него можно загружать любую независимую программу. Например, на SPARC-системах в качестве такой независимой программы можно, кроме ядра операционной системы, загружать диагностический монитор.

Если независимая программа является динамически компонуемым выполняемым файлом, команда **boot** загрузит интерпретатор (компоновщик/загрузчик), соответствующий формату выполняемого файла, и передаст управление этому командному интерпретатору. Если независимая программа скомпонована статически, она сразу выполняется.

Обычно в качестве программы загружается ядро. После загрузки ядро запускает систему UNIX, монтирует необходимые файловые системы (см. **vfstab(4)**) и загружает процесс **/sbin/init** для перевода системы в стандартное состояние ("**initdefault**"), указанное в файле **/etc/inittab**. См. **inittab(4)**.

Процедура начальной загрузки на платформе SPARC

На большинстве SPARC-машин процедура начальной загрузки состоит из следующих основных стадий.

После ключения питиания, *микропрограммное обеспечение системы* (firmware в PROM) выполняет ее тестирование (power-on self-test - POST). Форма и масштабы тестирования зависят от используемой в системе версии микропрограммного обеспечения.

После успешного выполнения тестирования, микропрограммное обеспечение пытается автоматически выполнить загрузку, если соответствующий флаг установлен в используемом ПЗУ. Можно изменять имя загружаемого файла и устройство, с которого он будет загружаться.

Эти флаги и имена можно установить с помощью команды **eeprom(1M)** из командного интерпретатора или с помощью команд PROM, задаваемых в ответ на приглашение **ok** после остановки системы.

В качестве вторичной загружаемой программы может использоваться либо программа **ufsboot** (при загрузке с диска), либо программа **inetboot** (при загрузке по сети). При загрузке с диска процесс начальной загрузки состоит из двух концептуально различных стадий: *первичной* и *вторичной* загрузки. PROM предполагает, что первичный загрузочный блок находится в блоках с 1 по 15 на локальном диске. При загрузке по сети PROM выполняет обратный запрос ARP и, при получении ответа, посылает широковещательный запрос по протоколу TFTP для получения программы **inetboot** по сети с любого ответившего сервера, и выполняет эту программу. Программа **inetboot** выполняет другой обратный запрос ARP, а затем использует протокол bootparams (см. **bootparams(4)**) для поиска корневой файловой системы. Затем она получает по сети ядро по протоколу NFS, и выполняет его.

Если имя независимой программы задано как относительное (не начинается с косой черты), вторичная программа загрузки будет использовать путь поиска, зависящий от платформы. Этот путь гарантированно включает каталог **/platform/имя_платформы**. На многих SPARC-платформах затем просматривается каталог **/platform/класс_платформы**. См. **filesystem(5)**. Если задано абсолютное имя, оно будет использовано при загрузке. Программа загрузки загружает соответствующую независимую программу и передает ей управление.

Если имя файла не задано в командной строке и не указано другим способом, например, в переменной **boot-file** NVRAM, программа **boot** выбирает для загрузки стандартный файл, соответствующий установленному в системе программному обеспечению, возможностям оборудования и микропрограммного обеспечения, а также сконфигурированному пользователем *файлу режима загрузки* (policy file).

Поведение команды boot OpenBoot PROM

Команда **boot** в OpenBoot принимает аргументы следующего вида:

```
ok boot [спецификация_устройства] [аргументы]
```

Стандартная команда **boot** без аргументов:

```
ok boot
```

Если **спецификация_устройства** в командной строке **boot** не задана, OpenBoot обычно использует значение переменной nvram **boot-device** или **diag-device**. Если в командной строке не заданы **аргументы**, OpenBoot обычно использует в качестве стандартных аргументов значения переменной nvram **boot-file** или **diag-file**. (Если система работает в режиме диагностики, вместо **boot-device** и **boot-file** используются переменные **diag-device** и **diag-file**).

Аргументы могут состоять из нескольких строк. Все строки аргментов передаются вторичной программе загрузки; они не интерпретируются системой OpenBoot.

Если в командной строке **boot** указаны аргументы, значения переменных nvram **boot-file** и **diag-file** не используются. Значения переменных nvram не объединяются с аргументами командной строки. Например, команда

```
ok boot -s
```

игнорирует значения переменных **boot-file** и **diag-file**; она интерпретирует строку "**-s**" как аргументы. **boot** не будет использовать значение переменной **boot-file** или **diag-file**.

Команды

```
ok boot net
```

и

```
ok boot cdrom
```

не имеют аргументов; они будут использовать значения переменных **boot-file** или **diag-file**, если они установлены, в качестве стандартного имени файла и аргументов для команды **boot**. Соответственно, если значением переменной **boot-file** является имя файла 64-битного ядра, и выполняется попытка загрузиться с установочного CD с помощью команды **boot cdrom**, загрузка завершится ошибкой, если на установочном CD находится только 32-битное ядро.

Поскольку значение переменной **boot-file** или **diag-file** может быть проигнорировано, в зависимости от вида используемой команды **boot**, в большинстве производственных систем нельзя полагаться на переменную **boot-file**. Чтобы изменить режим работы ОС, измените файл режима загрузки. Существенное исключение возникает лишь когда в производственной системе установлены как 32-битовые, так и 64-битовые пакеты, но использоваться должна 32-битовая версия ОС.

В большинстве случаев, лучше позволить команде **boot** выбрать соответствующее стандартное значение в зависимости от типа системы, ее аппаратного и микропрограммного обеспечения, а также содержимого корневой файловой системы. Общепринятой практикой является управление режимом работы команды **boot** путем изменения файла режима загрузки; однако, изменение значений переменных **boot-file** или **diag-file** может давать неожиданные результаты в некоторых ситуация.

Такое поведение характерно для большинства систем, основанных на версиях OpenBoot 2.x и 3.x. Учтите, что на некоторых платформах возможны отличия.

Процедура начальной загрузки на платформе IA

На системах IA процесс начальной загрузки состоит из двух принципиально различных стадий, первичной и вторичной загрузки. Первичная загрузка реализована в ПЗУ BIOS на системной плате и в расширениях BIOS в ПЗУ периферийных контроллеров. Она отличается тем, что позволяет управлять установленными периферийными устройствами и обеспечивает средства ввода/вывода через программные прерывания. Процесс загрузки начинается с загрузки первого физического сектора дискеты, жесткого диска или, если это поддерживает установленная в система версия BIOS, CD-ROM. Первичная программа загрузки реализована как код в реальном режиме процессора IA.

Вторичная программа загрузки загружается первичной. Она реализована как 32-битовый код в защищенном режиме. Она также загружает и использует специфические расширения BIOS периферийных устройств, написанные как код в реальном режиме процессора IA. Вторичная программа загрузки называется **boot.bin** и отвечает за чтение и загрузку с файловой системы UFS на жестком диске или CD, или загрузку по сети по протоколу NFS.

Вторичная программа загрузки запускает программу Configuration Assistant, которая определяет (возможно, с помощью пользователя), какие устройства установлены в системе. Вторичная программа загрузки затем читает сценарий в файле **/etc/bootrc**, который управляет процессом загрузки. Этот файл содержит команды загрузочного командного интерпретатора, описанные ниже, и может изменяться, чтобы адаптировать процесс загрузки к особенностям машины.

Стандартный сценарий **/etc/bootrc** предлагает пользователю ввести символ **b** для загрузки с указанными опциями, символ **i** для вызова интерпретатора в интерактивном режиме, или любой другой символ для загрузки стандартного ядра. После загрузки ядро запускает операционную систему, загружает необходимые модули, монтирует необходимые файловые системы (см. **vfstab(4)**) и запускаеет процесс **/sbin/init** для перевода системы в стандартное состояние ("**initdefault**"), указанное в файле **/etc/inittab**. См. **inittab(4)**.

ОПЦИИ

SPARC

имена_OBP
: Задают местонахождения open boot prom. Например, на рабочих станциях SPARC, местонахождение **/sbus/esp@0,800000/sd@3,0:a** обозначает SCSI-диск (**sd**) с идентификатором 3 и логическим номером (lun) 0 на шине SCSI, поддерживаемой контроллером (хост-адаптером) **esp**, подключенным к слоту 0.

файл
: Имя загружаемой независимой программы. Если имя файла явно не задано, либо в команднойстроке **boot**, либо в переменной NVRAM **boot-file**, программа **boot** выбирает соответствующее стандартное имя файла. В большинстве систем стандартное имя файла соответствует 32-битному ядру. В системах, которые могут работать и с 32-битными, и с 64-битными ядрами, приоритет имеет 64-битное ядро. Программа **boot** выбирает соответствующий стандартный файл для загрузки в зависимости от установленного в системе программного обеспечения, возможностей оборудования и микропрограммного обеспечения, а также конфигурируемого пользователем файла режима загрузки.

-a
: Программа **boot** интерпретирует этот флаг как требование запроса к пользователю и поэтому запрашивает имя независимой программы. Флаг '**-a**' затем передается независимой программе.

-f
: При загрузке системы Autoclient этот флаг заставляет программу **boot** игнорировать локальный кэш клиента и читать все файлы по сети с соответствующего файл-сервера. Этот флаг игнорируется для всех систем, не относящихся к категории Autoclient. Флаг **-f** передается затем независимой программе.

-V
: Выдает подробную отладочную информацию.

-D стандартный_файл
: Явно задает стандартный загружаемый файл. В некоторых системах программа **boot** выбирает стандартный файл динамически, если он не задан явно. Эта опция позволяет явно задать

стандартный файл и может пригодиться при загрузке отладчика ядра, **kadb(1M)**, поскольку по умолчанию **kadb** загружает значение переменной **default-file**, экспортированное программой **boot**.

флаги_загрузки
Программа **boot** передает все флаги загрузки указанному **файлу**. Они не интерпретируются программой **boot**. Подробнее об опциях стандартной загружаемой программы см. на страницах справочного руководства **kernel(1M)** и **kadb(1M)**.

аргументы_клиентской_программы
Программа **boot** передает все **аргументы_клиентской_программы** загружаемому **файлу**. Они не интерпретируются командой **boot**.

IA

файл
Имя загружаемой независимой программы. По умолчанию загружается файл / **platform/имя_платформы/kernel/unix** из корневого раздела, но можно задать в командной строке и другую программу.

-f
При загрузке системы Autoclient этот флаг заставляет программу **boot** игнорировать локальный кэш клиента и читать все файлы по сети с соответствующего файл-сервера. Этот флаг игнорируется для всех систем, не относящихся к категории Autoclient. Флаг **-f** передается затем независимой программе.

аргументы_загрузки
Программа **boot** передает все **аргументы_загрузки** загружаемому **файлу**. Они не интерпретируются командой **boot**. Подробнее об опциях ядра см. на страницах справочного руководства **kernel(1M)** и **kadb(1M)**.

ПОДРОБНОСТИ ПОСЛЕДОВАТЕЛЬНОСТИ ЗАГРУЗКИ В IA

После включения питания PC-совместимой машины микропрограммное обеспечение системы в ПЗУ BIOS выполняет самотестирование при включении (power-on self test - POST), запускает расширения BIOS ПЗУ периферийных контрорллеров и выполняет программное прерывание INT 19h, начальная загрузка. Обработчик прерывания INT 19h обычно выполняет стандартную процедуру начальной загрузки PC, которая состоит в попытке чтения первого физического сектора дискеты в первом дисководе или, если эта попытка завершится неудачно, первого жесткого диска. Затем процессор переходит на первый байт образа сектора в памяти.

Первичная загрузка IA

Первый сектор дискеты содержит главный загрузочный блок. Загрузочный блок отвечает за загрузку образа *начального загрузчика* (boot loader) **strap.com**, который затем загружает вторичную программу загрузки, **boot.bin**. Аналогичная последовательность действий выполняется и при загрузке с CD-ROM, но местонахождение и содержимое загрузочного блока определяется спецификацией El Torito. Загрузка El Torito тоже приводит к запуску программы **strap.com**, которая, в свою очередь, загружает **boot.bin**.

Первый сектор жесткого диска содержит главный загрузочный блок, в котором находится главная загрузочная программа и таблица разделов FDISK, названная так по имени программы PC, с помощью которой ее поддерживают. Главная загрузочная программа находит активный раздел в таблице FDISK, загружает его первый сектор и переходит на первый байт образа этого сектора в памяти. Этим завершается стандартная последовательность загрузки с жесткого диска на PC.

Раздел IA FDISK для программного обеспечения Solaris начинается с одноцилиндровой загрузочной секции (boot slice), содержащей программу загрузки раздела (**pboot**) в первом секторе, стандартную метку диска Solaris и *таблицу секций тома* (volume table of contents - VTOC) во втором и третьем секторах, и программу **bootblk** в четвертом и последующих секторах. Когда раздел FDISK для программного обеспечения Solaris является активным разделом, главная загрузочная программа (**mboot**) читает загрузочную программу раздела с первого сектора в память и передает ей управление. Эта программа, в свою очередь, считывает в память программу **bootblk** и передает управление ей. Независимо от типа активного раздела, если диск содержит несколько разделов FDISK, пользователь получает возможность загрузиться с любого раздела.

Программа **bootblk** или **strap.com** (в зависимости от типа активного раздела) читает программу **boot.bin** из файловой системы в корневой секции диска Solaris и переходит на первый байт ее образа в памяти.

Вторичная загрузка IA

Вторичная программа загрузки, **boot.bin**, переводит процессор в 32-разрядный защищенный режим со страничной организацией памяти и выполняет некоторые действия по инициализации системы. Она запускает программу Configuration Assistant, которая либо загружает систему автоматически, либо выдает список устройств, с которых можно загрузиться, в зависимости от значения переменной **auto-boot?** (см. **eeprom(1M)**).

Диски, с которых выполняется загрузка (в том числе, CDROM) должны содержать файловую систему UFS. При обращении к сетевым устройствам сначала посылается запрос Reverse Address Resolution Protocol (RARP) для получения IP-адреса машины, а затем RPC-вызов **bootparams** для поиска сервера, который может предоставить корневую файловую систему. Затем корневая файловая система монтируется по протоколу NFS. После успешного монтирования корневой файловой системы программа **boot.bin** вызывает специальный загрузочный командный интерпретатор, который интерпретирует содержимое файла **/etc/bootrc**.

Язык программирования вторичной загрузки для IA

Широкий спектр оборудования, которое необходимо поддерживать на IA-системах требует повышенной гибкости процесса загрузки. Эта гибкость частично достигается за счет возможности программировать процесс вторичной загрузки. Вторичная программа загрузки содержит интерпретатор, воспринимающий простой язык команд, похожий на языки командных интерпретаторов **sh** и **csh**. Основное отличие в том, что конвейеры, циклы и перенаправление ввода/вывода не поддерживается.

Лексемы

Загрузочный командный интерпретатор разбивает входные строки на слова, разделенные символами пробела и табуляции. Символы доллара (**$**), одиночной кавычки (**'**), двойной кавычки (**"**), номера (**#**), перевода строки и обратной косой (\) являются метасимволами. Чтобы использовать метасимволы буквально, их надо предварять обратной косой чертой. Символ новой строки, перед котым идет обратная косая, рассматривается как пробел. Символ номера начинает комментарий, который продолжается до символа новой строки.

Строка в одиночных или двойных кавычках образует одно слово (или часть одного слова. Пробелы и символы новой строки в кавычках становятся частью слова. Символы строки в кавычках можно маскировать, предваряя их символом обратной косой; таким образом, можно ввести одиночную кавычку в

строке в одиночных кавычках, если перед ней указать обратную косую. Две обратных косых подряд дают в результате обратную косую, а обратная косая перед симоволом новой строки позволяет включить символ новой строки в слово.

Переменные

По ходу загрузки поддерживается набор переменных со строковыми значениями. Первым символом имени переменной должна быть буква, а затем могут идти буквы, цифры и символы подчеркивания. Команда **set** создает переменную и/или присваивает ей значение, или выдает значения переменных. Команда **unset** удаляет переменную.

Подстановка значений переменных происходит, когда интерпретатор обнаруживает символ доллара, не предваренный обратной косой. Вместо доллара и имени переменной, которое за ним идет, подставляется значение этой переменной, и анализ строки продолжается с начала этого значения. Подстановка значений переменных выполняется также в строках в двойных кавычках, но не выполняется в строках в одиночных кавычках. Имя переменной может быть взято в фигурные скобки, отделяющие его от последующих символов.

Команды

Команда - это последовательность слов, завершаемая символом новой строки. Первое слово - имя команды, последующие слова - ее аргументы. Все команды являются встроенными. Независимые программы выполняются с помощью команды **run**.

Условное выполнение команд

Команды можно выполнять по условию, окружив командами **if**, **elseif**, **else** и **endif**:

```
if выражение1
    ...
elseif выражение2
    ...
elseif выражение3
    ...
else
    ...
endif
```

Блок **if ... endif** можно включать в другие блоки **if ... endif**.

Выражения

Команды **set**, **if** и **elseif** вычисляют арифметические выражения, синтаксис и семантика которых соответствует языку программирования C. Поддерживаются операторы ||, **&&**, |, ^, **&**, ==, **!=**, <, >, <=, >=, >>, <<, +, -, *, /, **%**, ~ и **!**, а также скобки **(**, **)** и оператор запятая. При вычислении используются 32-битовые целые числа со знаком.

Выражения анализируются после полного формирования командной строки. Каждая лексема в выражении должна представлять собой отдельное слово-аргумент, так что лексемы в командной строке надо разделять пробелами.

Перед выполнением арифметических операций над слолвом-операндом, оно преобразуется из строки в 32-битовое целочисленное значение со знаком. После необязательного знака начальный символ **0** означает преобразование из восьмеричной системы счисления, а начальные символы **0x** или **0X** требуют выполнить преобразование из шестнадцатеричной системы счисления. В противном случае предполагается, что строка задает десятичное значение. Строка, не допускающая преобразования в целое число, заменяется нулем.

Предлагается ряд функций обработки строк. Имена встроенных функций начинаются с точки. Строковые аргументы этих функций не преобразуются в целые числа. Чтобы оператор, например, **-**, обрабатывался как строка, его необходимо предварять обратной косой, а эта обратная косая должна маскироваться другой обратной косой. Учтите также, что пустая строка может дать в результате пустой аргумент, и, как следствие, синтаксическую ошибку в выражении. Например:

```
if .strneq ( ${usrarg}X , \- , 1 )
```

Это безопасный способ проверить, что значение переменной **usrarg** начинается с **-**, даже если оно может быть пустым.

Ввод/вывод

Загрузочный командный интерпретатор принимает ввод с консоли или из файлов. Команда **source** требует от интерпретатора прочитать файл в память и начать его анализ. Команда **console** требует от интерррпетатора читать входные данные с системной консоли. При прочтении символа конца файла (EOF) интерпретатор продолжает разбор прежнего источника входных данных. Символ CTRL-D, введенный в начале строки с консоли, рассматривается как признак конца файла (EOF).

Команда **echo** выдает свои аргументы на экран. Команда **read** читает данные с системной консоли и присваивает значения обнаруженных в ней слов своим аргументам-переменным.

Отладка

Команда **verbose** включает и отключает режим подробного информирования. В режиме подробного информирования интерпретатор выдает строки из текущего исходного файла и команды в том виде, как они будут выполняться после подстановки значений переменных.

Команда **singlestep** включает и отключает режим пошагового выполнения команд. В режиме пошагового выполнения командный интерпретатор выдает **step ?** перед обработкой следующей команды и ждет ввода с клавиатуры, который не запоминается. Обработка начинается при нажатии клавиши ENTER. Это позволяет пошагово выполнять команды в режиме подробного информирования.

Инициализация

При первоначальном вызове интерпретатора в процессе загрузки он начинает работу с выполнения встроенной строки инициализации. Эта строка обычно имеет вид "**source /etc/bootrc\n**" и предназначена для запуска сценария загрузки с корневой файловой системы.

Взаимодействие с другими независимыми программами

Загрузочный интерпретатор передает информацию другим независимым программам с помощью аргументов команды **run**. Программа может вернуть информацию загрузочному интерпретатору, устанавливая значения переменных интерпретатора с помощью служебной функции загрузки **var_ops()**. Она также может передавать информацию ядру с помощью служебной функции загрузки **setprop()**. Свойство **whoami** устанавливается равным имени выполняющейся независимой программы.

Встроенные команды

console
Интерпретирует ввод с консоли вплоть до символа CTRL-D.

echo аргумент1 ...
Выдает аргументы через пробел, завершая символом новой строки.

echo -n аргумент1 ...
Выдает аргументы через пробел, не завершая символом новой строки.

getprop свойство переменная
Присваивает значение **свойства** указанной **переменной**. Значение свойства длиной ноль дает пустую строку. Если свойство не существует, переменная не устанавливается.

getproplen свойство переменная
Присваивает длину значения **свойства** в виде шестнадцатеричного числа указанной **переменной**. Длина значения свойства включает завершающий 0. Если **свойство** не существует, переменная получает значение **0xFFFFFFFF** (**-1**).

if выражение
Если **выражение** истинно, выполняет инструкции вплоть до следующей встроенной команды **elseif**, **else** или **endif**. Если **выражение** ложно, не выполняет эти команды.

elseif выражение
Если все предыдущие команды **if** и **elseif** не сработали и **выражение** истинно, выполняет инструкции вплоть до следующей встроенной команды **elseif**, **else** или **endif**. В противном случае, не выполняет эти команды.

else
Если все предыдущие команды **if** и **elseif** не сработали и **выражение** истинно, выполняет инструкции вплоть до следующей встроенной команды **endif**. В противном случае, не выполняет эти команды.

endif
Возвращает управление окружающему блоку.

help
Выдает экран справки, содержащий резюме по всем командам загрузочного командного интерпретатора.

read имя1 ...
Читает строку с консоли, разбивает ее на слова и присваивает их как значения указанным переменным **имя1** и т.д.

readt ожидание ...
: То же, что и **read**, но завершается, если строка не была введена в течение указанного количества секунд **ожидания**.

run программа аргумент1 ...
: Загружает и запускает независимую **программу**, передавая ей **аргумент1** и другие аргументы.

set
: Выдает все установденные переменные с их значениями.

set переменная
: Устанавливает указанной **переменной** значение пустая строка.

set переменная слово
: Устанавливает указанной **переменной** значение **слово**.

set переменная выражение
: Устанавливает **переменной** значение **выражения**. **Выражение** должно состоять из нескольких слов. Значение преобразуется в шестнадцатеричное без знака, так что **-1** представляется как **0xFFFFFFFF**.

setcolor
: Устанавливает атрибуты текстового режима дисплея. Допускаются цвета **black**, **blue**, **green**, **cyan**, **red**, **magenta**, **brown**, **white**, **gray**, **lt_blue**, **lt_green**, **lt_cyan**, **lt_red**, **lt_magenta**, **yellow** и **hi_white**.

setprop свойство слово
: Устанавливает указанному **свойству** значение **слово**.

singlestep или **singlestep on**
: Включает режим пошагового выполнения, в котором интерпретатор выдает приглашение **step ?** перед обработкой каждой команды, и ждет ввода с клавиатуры. Для выполнения следующей команды нажмите клавишу ENTER.

singlestep off
: Отключает режим пошагового выполнения.

source файл
: Считывает **файл** в память и начинает его интерпретировать. При достижении символа конца файла (EOF), возвращается к предыдущему источнику входных данных.

unset переменная
: Удаляет указанную **переменную**.

verbose или **verbose on**
: Включает режим подробного информирования, в котором выдаются на экран строки из входных файлов и выполняемые команды.

verbose off
: Отключает режим подробного информирования.

Встроенные функции

В выражениях поддерживаются следующие встроенные функции:

.strcmp(строка1, строка2)
: Возвращает целочисленное значение, меньшее, равное или большее нуля в зависимости от того, является ли **строка1**, соответственно, лексикографически меньшей, равной или большей, чем **строка2**.

.strncmp(строка1, строка2, n)

Возвращает целочисленное значение, меньшее, равное или большее нуля в зависимости от того, является ли **строка1**, соответственно, лексикографически меньшей, равной или большей, чем **строка2**. Сравнивается не более **n** первых символов строк.

.streq (строка1, строка2)

Возвращает истину, если **строка1** равна **строке2**, и ложь в противном случае.

.strneq (строка1, строка2, n)

Возвращает истину, если **строка1** равна **строке2**, и ложь в противном случае. Сравнивается не более **n** первых символов строк.

.strfind (строка, адрес, n)

Просматривает **n** адресов в памяти, начиная с указанного **адреса** в поисках начала **строки**. Строка в памяти не обязательно должна заканчиваться нулевым байтом. Возвращает истину, если **строка** найдена, и ложь в противном случае. Функцию **.strfind** можно использовать для поиска строк в ПЗУ BIOS и расширениях BIOS, идентифицирующих машины и контроллеры периферийных устройств.

ПРИМЕРЫ

SPARC

Пример 1: загрузка стандартного ядра в однопользовательском интерактивном режиме

Для загрузки стандартного ядра в однопользовательском интерактивном режиме, введите одну из следующих команд в ответ на приглашение **ok**:

```
boot -as
```

```
boot disk3 -as
```

32-битовый SPARC

Пример 2: Загрузка отладчика kadb при указании 32-битового ядра в качестве стандартного файла

Для загрузки **kadb** для отладки 32-битового ядра:

```
boot kadb -D kernel/unix
```

Пример 3: явная загрузка 32-битового ядра

Для явной загрузки 32-битового ядра необходимо указать имя файла ядра. Поэтому, чтобы загрузить 32-битовое ядро в однопользовательском интерактивном режиме, введите одну из следующих команд в ответ на приглашение **ok**:

```
boot kernel/unix -as
```

```
boot disk3 kernel/unix -as
```

64-битовый SPARC

Пример 4: явная загрузка 64-битового ядра

Для явной загрузки 64-битового ядра необходимо указать имя файла ядра. Поэтому, чтобы загрузить 64-битовое ядро в однопользовательском интерактивном режиме, введите одну из следующих команд в ответ на приглашение **ok**:

```
boot kernel/sparcv9/unix -as

boot disk3 kernel/sparcv9/unix -as
```

Прежде чем явно загружать 64-битовое ядро, указав имя соответствующего файла, прочитайте подраздел "**Загрузка систем UltraSPARC**" в разделе "**ПРИМЕЧАНИЯ**".

IA

Пример 5: загрузка стандартного ядра в однопользовательском интерактивном режиме

Для загрузка стандартного ядра в однопользовательском интерактивном режиме, введите одну из следующих команд в ответ на приглашение >:

```
b -as

b kernel/unix -as
```

ФАЙЛЫ

/platform/имя_платформы/ufsboot
: программа вторичной загрузки с диска или CD.

/etc/inittab
: таблица, в которой задано значение стандартного состояния "**initdefault**".

/sbin/init
: программа, переводящая систему в состояние "**initdefault**".

/platform/имя_платформы/boot.conf
/platform/имя_класса_оборудования/boot.conf
: Основной и альтернативный пути поиска файла режима загрузки. Учтите, что файл режима загрузки реализован реализован не на всех платформах.

32-битовый SPARC и IA

/platform/имя_платформы/kernel/unix
: стандартная программа загрузки системы (ядро).

Только 64-битовый SPARC

/platform/имя_платформы/kernel/sparcv9/unix
: стандартная программа загрузки системы (ядро).

См. подраздел "**Загрузка систем UltraSPARC**" в разделе "**ПРИМЕЧАНИЯ**".

Только IA

/etc/bootrc

сценарий, управляющий процессом загрузки.

/platform/имя_платформы/boot/solaris/boot.bin

вторичная программа загрузки, используемая в системах IA вместо **ufsboot**.

/platform/имя_платформы/boot

каталог, содержащий файлы, связанные с процессом загрузки.

ССЫЛКИ

uname(1), **eeprom(1M)**, **init(1M)**, **installboot(1M)**, **kadb(1M)**, **kernel(1M)**, **shutdown(1M)**, **uadmin (2)**, **bootparams(4)**, **inittab(4)**, **vfstab(4)**, **filesystem(5)**

System Administration Guide, Volume 1
Sun Hardware Platform Guide

ПРЕДУПРЕЖДЕНИЯ

Утилита **boot** не способна определить, какие файлы можно использовать в качестве загружаемых программ. Если затребована загрузка файла, которые не является загружаемым, утилита **boot** загрузит его и передаст ему управление. Дальнейшие события непредсказуемы.

ПРИМЕЧАНИЯ

Имя_платформы можно узнать с помощью опции **-i** команды **uname(1)**. **Имя_класса_оборудования** можно узнать с помощью опции **-m** команды **uname(1)**.

64-битовый SPARC
Загрузка систем UltraSPARC

На некоторых платформах для обеспечения работы 64-битового ядра может потребоваться обновление микропрограммного обеспечения. Подробнее см. в **Sun Hardware Platform Guide**. Если пакеты 64-битового ядра установлены и **boot** определяет, что необходимо обновление микропрограммного обеспечения для работы в 64-битовом режиме, **boot** выдает на консоль соответствующее сообщение и использует в качестве стандартного файла 32-битовое ядро.

В системах, сожержащих процессоры UltraSPARC-1 с тактовой частотой до 200 МГц включительно, пользователь может запустить 64-битовую программу, которая использует имеющуюся аппаратную проблему для остановки процессора. Поскольку 64-битовые программы не могут работать на 32-битовом ядре, в качестве стандартного файла в этих системах выбирается 32-битовое ядро.

Последовательность команд, вызвающая аппаратную проблему, очень нетипична, и маловероятно, что она будет сгенерирована компилятором. Для демонстрации проблемы необходимо специально писать код на ассемблере. Очень маловероятно, что написанная на ассемблере по всем правилам вручную процедура будет включать эту последовательность команд.

Пользователи, согласные почти на риск случайного или намеренного запуска пользователем программы, специально созданной для остановки процессора, могут запускать 64-битовое ядро, изменив файл режима

загрузки. Отредактируйте файл **/platform/имя_платформы/boot.conf** так, чтобы он содержал незакомментированную строку с установкой переменной **ALLOW_64BIT_KERNEL_ON_UltraSPARC_1_CPU** значение **true**, как показано ниже:

```
ALLOW_64BIT_KERNEL_ON_UltraSPARC_1_CPU=true
```

Подробнее об этом см. в руководстве **Sun Hardware Platform Guide**.

Только IA

Поскольку клавиша "-" на клавиатурах с национальными раскладками перенесена, необходимо использовать альтернативную клавишу для передачи аргументов команде загрузки в системах на основе IA, использующих такие клавиатуры. Используйте "-" на числовой клавиатуре. Альтернативные клавиши, которые можно нажимать на таких "национальных" клавиатурах для ввода символа "-" в процессе загрузки представлены ниже.

Клавиатура	**Замещающая клавиша**
итальянская	**'**
испанская	**'**
шведская	+
французская	**?**
немецкая	**?**

Например, команду **b -r** необходимо на шведскойм клавиатуре набирать как **b +r**, - на экране при этом появится **b -r**.

Последнее изменение: 26 февраля 1999 года