NAME

loader_simp - kernel bootstrapping final stage

DESCRIPTION

The program called **loader_simp** is the final stage of FreeBSD's kernel bootstrapping process. On IA32 (i386) architectures, it is a *BTX* client. It is linked statically to libsa(3) and usually located in the directory */boot*.

It provides a scripting language that can be used to automate tasks, do pre-configuration or assist in recovery procedures. This scripting language is roughly divided in two main components. The smaller one is a set of commands designed for direct use by the casual user, called "builtin commands" for historical reasons. The main drive behind these commands is user-friendliness.

During initialization, **loader_simp** will probe for a console and set the *console* variable, or set it to serial console ("comconsole") if the previous boot stage used that. If multiple consoles are selected, they will be listed separated by spaces. Then, devices are probed, *currdev* and *loaddev* are set, and *LINES* is set to 24. After that, */boot/loader.rc* is processed if available. These files are processed through the **include** command, which reads all of them into memory before processing them, making disk changes possible.

At this point, if an **autoboot** has not been tried, and if *autoboot_delay* is not set to "NO" (not case sensitive), then an **autoboot** will be tried. If the system gets past this point, *prompt* will be set and **loader_simp** will engage interactive mode. Please note that historically even when *autoboot_delay* is set to "0" user will be able to interrupt autoboot process by pressing some key on the console while kernel and modules are being loaded. In some cases such behaviour may be undesirable, to prevent it set *autoboot_delay* to "-1", in this case **loader_simp** will engage interactive mode only if **autoboot** has failed.

BUILTIN COMMANDS

In **loader_simp**, builtin commands take parameters from the command line. Presently, the only way to call them from a script is by using *evaluate* on a string. In the case of an error, an error message will be displayed and the interpreter's state will be reset, emptying the stack and restoring interpreting mode.

The builtin commands available are:

autoboot [*seconds* [*prompt*]]

Proceeds to bootstrap the system after a number of seconds, if not interrupted by the user. Displays a countdown prompt warning the user the system is about to be booted, unless interrupted by a key press. The kernel will be loaded first if necessary. Defaults to 10 seconds.

bcachestat

Displays statistics about disk cache usage. For debugging only.

boot
boot *kernelname* [**...**]
boot -flag ...
Immediately proceeds to bootstrap the system, loading the kernel if necessary. Any flags or arguments are passed to the kernel, but they must precede the kernel name, if a kernel name is provided.

echo [**-n**] [<message>]
Displays text on the screen. A new line will be printed unless **-n** is specified.

heap Displays memory usage statistics. For debugging purposes only.

help [topic [subtopic]]
Shows help messages read from */boot/loader.help*. The special topic *index* will list the topics available.

include *file* [*file* ...]
Process script files. Each file, in turn, is completely read into memory, and then each of its lines is passed to the command line interpreter. If any error is returned by the interpreter, the include command aborts immediately, without reading any other files, and returns an error itself (see *ERRORS*).

load [**-t** *type*] *file* **...**
Loads a kernel, kernel loadable module (kld), disk image, or file of opaque contents tagged as being of the type *type*. Kernel and modules can be either in a.out or ELF format. Any arguments passed after the name of the file to be loaded will be passed as arguments to that file. Use the md_image type to make the kernel create a file-backed md(4) disk. This is useful for booting from a temporary rootfs. Currently, argument passing does not work for the kernel.

load_geli [**-n** *keyno*] *prov file*
Loads a geli(8) encryption keyfile for the given provider name. The key index can be specified via *keyno* or will default to zero.

ls [**-l**] [*path*]
Displays a listing of files in the directory *path*, or the root directory if *path* is not specified. If **-l** is specified, file sizes will be shown too.

lsdev [**-v**]

Lists all of the devices from which it may be possible to load modules, as well as ZFS pools. If **-v** is specified, more details are printed, including ZFS pool information in a format that resembles **zpool status** output.

lsmod [**-v**]
Displays loaded modules. If **-v** is specified, more details are shown.

lszfs *filesystem*
A ZFS extended command that can be used to explore the ZFS filesystem hierarchy in a pool. Lists the immediate children of the *filesystem*. The filesystem hierarchy is rooted at a filesystem with the same name as the pool.

more *file* [*file* ...]
Display the files specified, with a pause at each *LINES* displayed.

pnpscan [**-v**]
Scans for Plug-and-Play devices. This is not functional at present.

read [**-t** *seconds*] [**-p** *prompt*] [*variable*]
Reads a line of input from the terminal, storing it in *variable* if specified. A timeout can be specified with **-t**, though it will be canceled at the first key pressed. A prompt may also be displayed through the **-p** flag.

reboot Immediately reboots the system.

set *variable*
set *variable=value*
Set loader's environment variables.

show [*variable*]
Displays the specified variable's value, or all variables and their values if *variable* is not specified.

unload Remove all modules from memory.

unset *variable*
Removes *variable* from the environment.

? Lists available commands.

BUILTIN ENVIRONMENT VARIABLES

Environment variables can be set and unset through the **set** and **unset** builtins, and can have their values interactively examined through the use of the **show** builtin. Their values can also be accessed as described in *BUILTIN PARSER*.

Notice that these environment variables are not inherited by any shell after the system has been booted.

A few variables are set automatically by **loader_simp**. Others can affect the behavior of either **loader_simp** or the kernel at boot. Some options may require a value, while others define behavior just by being set. Both types of builtin variables are described below.

autoboot_delay

Number of seconds **autoboot** will wait before booting. Configuration options are described in loader.conf(5).

boot_askname

Instructs the kernel to prompt the user for the name of the root device when the kernel is booted.

boot_cdrom

Instructs the kernel to try to mount the root file system from CD-ROM.

boot_ddb

Instructs the kernel to start in the DDB debugger, rather than proceeding to initialize when booted.

boot_dfltroot

Instructs the kernel to mount the statically compiled-in root file system.

boot_gdb

Selects gdb-remote mode for the kernel debugger by default.

boot_multicons

Enables multiple console support in the kernel early on boot. In a running system, console configuration can be manipulated by the conscontrol(8) utility.

boot_mute

All kernel console output is suppressed when console is muted. In a running system, the state of console muting can be manipulated by the conscontrol(8) utility.

boot_pause
: During the device probe, pause after each line is printed.

boot_serial
: Force the use of a serial console even when an internal console is present.

boot_single
: Prevents the kernel from initiating a multi-user startup; instead, a single-user mode will be entered when the kernel has finished device probing.

boot_verbose
: Setting this variable causes extra debugging information to be printed by the kernel during the boot phase.

bootfile List of semicolon-separated search path for bootable kernels. The default is "kernel".

comconsole_speed
: Defines the speed of the serial console (i386 and amd64 only). If the previous boot stage indicated that a serial console is in use then this variable is initialized to the current speed of the console serial port. Otherwise it is set to 115200 unless this was overridden using the *BOOT_COMCONSOLE_SPEED* variable when **loader_simp** was compiled. Changes to the *comconsole_speed* variable take effect immediately.

comconsole_port
: Defines the base i/o port used to access console UART (i386 and amd64 only). If the variable is not set, its assumed value is 0x3F8, which corresponds to PC port COM1, unless overridden by *BOOT_COMCONSOLE_PORT* variable during the compilation of **loader_simp**. Setting the *comconsole_port* variable automatically set *hw.uart.console* environment variable to provide a hint to kernel for location of the console. Loader console is changed immediately after variable *comconsole_port* is set.

comconsole_pcidev
: Defines the location of a PCI device of the 'simple communication' class to be used as the serial console UART (i386 and amd64 only). The syntax of the variable is 'bus:device:function[:bar]', where all members must be numeric, with possible 0x prefix to indicate a hexadecimal value. The *bar* member is optional and assumed to be 0x10 if omitted. The bar must decode i/o space. Setting the variable *comconsole_pcidev* automatically sets the variable *comconsole_port* to the base of the selected bar, and hint *hw.uart.console*. Loader console is changed immediately after variable *comconsole_pcidev* is set.

console Defines the current console or consoles. Multiple consoles may be specified. In that case, the first listed console will become the default console for userland output (e.g. from init(8)).

currdev Selects the default device to loader the kernel from. The syntax is:

loader_device:

or

zfs:dataset:

Examples:

disk0p2:

zfs:zroot/ROOT/default:

dumpdev

Sets the device for kernel dumps. This can be used to ensure that a device is configured before the corresponding *dumpdev* directive from rc.conf(5) has been processed, allowing kernel panics that happen during the early stages of boot to be captured.

init_chroot

See init(8).

init_exec

See init(8).

init_path

Sets the list of binaries which the kernel will try to run as the initial process. The first matching binary is used. The default list is "/sbin/init:/sbin/oinit:/sbin/init.bak:/rescue/init".

init_script

See init(8).

init_shell

See init(8).

interpret

Has the value "OK" if the Forth's current state is interpreting.

LINES Define the number of lines on the screen, to be used by the pager.

module_path

Sets the list of directories which will be searched for modules named in a load command or implicitly required by a dependency. The default value for this variable is

"/boot/kernel;/boot/modules".

num_ide_disks
Sets the number of IDE disks as a workaround for some problems in finding the root disk at boot. This has been deprecated in favor of *root_disk_unit*.

prompt Value of **loader_simp**'s prompt. Defaults to "${interpret}". If variable *prompt* is unset, the default prompt is '>'.

root_disk_unit
If the code which detects the disk unit number for the root disk is confused, e.g. by a mix of SCSI and IDE disks, or IDE disks with gaps in the sequence (e.g. no primary slave), the unit number can be forced by setting this variable.

rootdev By default the value of *currdev* is used to set the root file system when the kernel is booted. This can be overridden by setting *rootdev* explicitly.

Other variables are used to override kernel tunable parameters. The following tunables are available:

efi.rt.disabled Disable UEFI runtime services in the kernel, if applicable. Runtime services are only available and used if the kernel is booted in a UEFI environment.

hw.physmem Limit the amount of physical memory the system will use. By default the size is in bytes, but the **k**, **K**, **m**, **M**, **g** and **G** suffixes are also accepted and indicate kilobytes, megabytes and gigabytes respectively. An invalid suffix will result in the variable being ignored by the kernel.

hw.pci.host_start_mem, *hw.acpi.host_start_mem*
When not otherwise constrained, this limits the memory start address. The default is 0x80000000 and should be set to at least size of the memory and not conflict with other resources. Typically, only systems without PCI bridges need to set this variable since PCI bridges typically constrain the memory starting address (and the variable is only used when bridges do not constrain this address).

hw.pci.enable_io_modes
Enable PCI resources which are left off by some BIOSes or are not enabled correctly by the device driver. Tunable value set to ON (1) by default, but this may cause problems with some peripherals.

kern.maxusers Set the size of a number of statically allocated system tables; see tuning(7) for a

description of how to select an appropriate value for this tunable. When set, this tunable replaces the value declared in the kernel compile-time configuration file.

kern.ipc.nmbclusters

Set the number of mbuf clusters to be allocated. The value cannot be set below the default determined when the kernel was compiled.

kern.ipc.nsfbufs

Set the number of sendfile(2) buffers to be allocated. Overrides NSFBUFS. Not all architectures use such buffers; see sendfile(2) for details.

kern.maxswzone

Limits the amount of KVM to be used to hold swap metadata, which directly governs the maximum amount of swap the system can support, at the rate of approximately 200 MB of swap space per 1 MB of metadata. This value is specified in bytes of KVA space. If no value is provided, the system allocates enough memory to handle an amount of swap that corresponds to eight times the amount of physical memory present in the system.

Note that swap metadata can be fragmented, which means that the system can run out of space before it reaches the theoretical limit. Therefore, care should be taken to not configure more swap than approximately half of the theoretical maximum.

Running out of space for swap metadata can leave the system in an unrecoverable state. Therefore, you should only change this parameter if you need to greatly extend the KVM reservation for other resources such as the buffer cache or *kern.ipc.nmbclusters*. Modifies kernel option VM_SWZONE_SIZE_MAX.

kern.maxbcache

Limits the amount of KVM reserved for use by the buffer cache, specified in bytes. The default maximum is 200MB on i386, and 400MB on amd64. This parameter is used to prevent the buffer cache from eating too much KVM in large-memory machine configurations. Only mess around with this parameter if you need to greatly extend the KVM reservation for other resources such as the swap zone or *kern.ipc.nmbclusters*. Note that the NBUF parameter will override this limit. Modifies VM_BCACHE_SIZE_MAX.

kern.msgbufsize

Sets the size of the kernel message buffer. The default limit of 96KB is usually sufficient unless large amounts of trace data need to be collected between opportunities

to examine the buffer or dump it to a file. Overrides kernel option MSGBUF_SIZE.

machdep.disable_mtrrs
Disable the use of i686 MTRRs (x86 only).

net.inet.tcp.tcbhashsize
Overrides the compile-time set value of TCBHASHSIZE or the preset default of 512. Must be a power of 2.

twiddle_divisor
Throttles the output of the 'twiddle' I/O progress indicator displayed while loading the kernel and modules. This is useful on slow serial consoles where the time spent waiting for these characters to be written can add up to many seconds. The default is 16; a value of 32 spins half as fast, while a value of 8 spins twice as fast.

vm.kmem_size
Sets the size of kernel memory (bytes). This overrides the value determined when the kernel was compiled. Modifies VM_KMEM_SIZE.

vm.kmem_size_min

vm.kmem_size_max
Sets the minimum and maximum (respectively) amount of kernel memory that will be automatically allocated by the kernel. These override the values determined when the kernel was compiled. Modifies VM_KMEM_SIZE_MIN and VM_KMEM_SIZE_MAX.

ZFS FEATURES

loader_simp supports the following format for specifying ZFS filesystems which can be used wherever loader(8) refers to a device specification:

zfs:pool/filesystem:

where *pool/filesystem* is a ZFS filesystem name as described in zfs(8).

If */etc/fstab* does not have an entry for the root filesystem and *vfs.root.mountfrom* is not set, but *currdev* refers to a ZFS filesystem, then **loader_simp** will instruct kernel to use that filesystem as the root filesystem.

SECURITY

Access to the **loader_simp** command line provides several ways of compromising system security, including, but not limited to:

- Booting from removable storage.

  One can prevent unauthorized access to the **loader_simp** command line by booting unconditionally in *loader.rc*. In order for this to be effective, one should also configure the firmware (BIOS or UEFI) to prevent booting from unauthorized devices.

FILES

/boot/loader_simp **loader_simp** itself.
/boot/loader.rc The script run by **loader_simp** on startup.

EXAMPLES

Boot in single user mode:

```
boot -s
```

Load the kernel, a splash screen, and then autoboot in five seconds. Notice that a kernel must be loaded before any other **load** command is attempted.

```
load kernel
load splash_bmp
load -t splash_image_data /boot/chuckrulez.bmp
autoboot 5
```

Set the disk unit of the root device to 2, and then boot. This would be needed in a system with two IDE disks, with the second IDE disk hardwired to ada2 instead of ada1.

```
set root_disk_unit=2
boot /boot/kernel/kernel
```

Set the default device used for loading a kernel from a ZFS filesystem:

```
set currdev=zfs:tank/ROOT/knowngood:
```

ERRORS

The following values are thrown by **loader_simp**:

100 Any type of error in the processing of a builtin.

-1 **Abort** executed.

-2 **Abort"** executed.

-56 **Quit** executed.

-256 Out of interpreting text.

-257 Need more text to succeed -- will finish on next run.

-258 **Bye** executed.

-259 Unspecified error.

SEE ALSO

libsa(3), loader.conf(5), tuning(7), boot(8), btxld(8)

HISTORY

The **loader_simp** first appeared in FreeBSD 3.1.

AUTHORS

The **loader_simp** was written by Michael Smith <msmith@FreeBSD.org>.