NAME

loader_4th - kernel bootstrapping final stage

DESCRIPTION

The program called **loader_4th** 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. The bigger component is an ANS Forth compatible Forth interpreter based on FICL, by John Sadler.

During initialization, **loader_4th** 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. Next, FICL is initialized, the builtin words are added to its vocabulary, and */boot/boot.4th* is processed if it exists. No disk switching is possible while that file is being read. The inner interpreter **loader_4th** will use with FICL is then set to **interpret**, which is FICL’s default. 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_4th** 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_4th** will engage interactive mode only if **autoboot** has failed.

BUILTIN COMMANDS

In **loader_4th**, 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. If an error condition occurs, an exception will be generated, which can be intercepted using ANS Forth exception handling words. If not intercepted, an error message will be displayed and the interpreter’s state will be reset, emptying the stack and restoring interpreting mode. The commands are described in the loader_simp(8) "BUILTIN COMMANDS" section.

BUILTIN ENVIRONMENT VARIABLES

The environment variables common to all interpreters are described in the loader_simp(8) "BUILTIN

ENVIRONMENT VARIABLES" section.

BUILTIN PARSER

When a builtin command is executed, the rest of the line is taken by it as arguments, and it is processed by a special parser which is not used for regular Forth commands.

This special parser applies the following rules to the parsed text:

1. All backslash characters are preprocessed.

   - \b , \f , \r , \n and \t are processed as in C.
   - \s is converted to a space.
   - \v is converted to ASCII 11.
   - \z is just skipped. Useful for things like "\0xf\z\0xf".
   - \0xN and \0xNN are replaced by the hex N or NN.
   - \NNN is replaced by the octal NNN ASCII character.
   - \" , \' and \$ will escape these characters, preventing them from receiving special treatment in Step 2, described below.
   - \\ will be replaced with a single \ .
   - In any other occurrence, backslash will just be removed.

2. Every string between non-escaped quotes or double-quotes will be treated as a single word for the purposes of the remaining steps.

3. Replace any $VARIABLE or ${VARIABLE} with the value of the environment variable *VARIABLE*.

4. Space-delimited arguments are passed to the called builtin command. Spaces can also be escaped through the use of \\ .

An exception to this parsing rule exists, and is described in *BUILTINS AND FORTH*.

BUILTINS AND FORTH

All builtin words are state-smart, immediate words. If interpreted, they behave exactly as described previously. If they are compiled, though, they extract their arguments from the stack instead of the command line.

If compiled, the builtin words expect to find, at execution time, the following parameters on the stack:

addrN lenN ... addr2 len2 addr1 len1 N

where *addrX lenX* are strings which will compose the command line that will be parsed into the builtin's arguments. Internally, these strings are concatenated in from 1 to N, with a space put between each one.

If no arguments are passed, a 0 *must* be passed, even if the builtin accepts no arguments.

While this behavior has benefits, it has its trade-offs. If the execution token of a builtin is acquired (through **'** or **[']**), and then passed to **catch** or **execute**, the builtin behavior will depend on the system state *at the time* **catch** *or* **execute** *is processed!* This is particularly annoying for programs that want or need to handle exceptions. In this case, the use of a proxy is recommended. For example:

```
: (boot) boot;
```

FICL

FICL is a Forth interpreter written in C, in the form of a forth virtual machine library that can be called by C functions and vice versa.

In **loader_4th**, each line read interactively is then fed to FICL, which may call **loader_4th** back to execute the builtin words. The builtin **include** will also feed FICL, one line at a time.

The words available to FICL can be classified into four groups. The ANS Forth standard words, extra FICL words, extra FreeBSD words, and the builtin commands; the latter were already described. The ANS Forth standard words are listed in the *STANDARDS* section. The words falling in the two other groups are described in the following subsections.

FICL EXTRA WORDS

.env

.ver

-roll

2constant

>name

body>

compare This is the STRING word set’s **compare**.

compile-only

endif

forget-wid

parse-word

sliteral This is the STRING word set’s **sliteral**.

wid-set-super

w@

w!

x.

empty

cell-

-rot

FREEBSD EXTRA WORDS

$ (--) Evaluates the remainder of the input buffer, after having printed it first.

% (--) Evaluates the remainder of the input buffer under a **catch** exception guard.

.# Works like . but without outputting a trailing space.

fclose (*fd* --) Closes a file.

fkey (*fd* -- *char*)
Reads a single character from a file.

fload (*fd* --) Processes a file *fd*.

fopen (*addr len mode* -- *fd*)
Opens a file. Returns a file descriptor, or -1 in case of failure. The *mode* parameter selects whether the file is to be opened for read access, write access, or both. The constants O_RDONLY, O_WRONLY, and O_RDWR are defined in */boot/support.4th*, indicating read only, write only, and read-write access, respectively.

fread (*fd addr len* -- *len'*)
Tries to read *len* bytes from file *fd* into buffer *addr*. Returns the actual number of bytes read, or -1 in case of error or end of file.

heap? (-- *cells*)
Return the space remaining in the dictionary heap, in cells. This is not related to the heap used by dynamic memory allocation words.

inb (*port* -- *char*)
Reads a byte from a port.

key (-- *char*) Reads a single character from the console.

key? (-- *flag*) Returns **true** if there is a character available to be read from the console.

ms (*u* --) Waits *u* microseconds.

outb (*port char* --)
Writes a byte to a port.

seconds (-- *u*) Returns the number of seconds since midnight.

tib> (-- *addr len*)
Returns the remainder of the input buffer as a string on the stack.

trace! (*flag* --) Activates or deactivates tracing. Does not work with **catch**.

FREEBSD DEFINED ENVIRONMENTAL QUERIES

arch-i386
TRUE if the architecture is IA32.

FreeBSD_version

FreeBSD version at compile time.

loader_version
: **loader_4th** version.

SECURITY

Access to the **loader_4th** command line provides several ways of compromising system security, including, but not limited to:

- Booting from removable storage, by setting the *currdev* or *loaddev* variables
- Executing binary of choice, by setting the *init_path* or *init_script* variables
- Overriding ACPI DSDT to inject arbitrary code into the ACPI subsystem

One can prevent unauthorized access to the **loader_4th** command line by setting the *password*, or setting *autoboot_delay* to -1. See loader.conf(5) for details. In order for this to be effective, one should also configure the firmware (BIOS or UEFI) to prevent booting from unauthorized devices.

MD

Memory disk (MD) can be used when the **loader_4th** was compiled with *MD_IMAGE_SIZE*. The size of the memory disk is determined by *MD_IMAGE_SIZE*. If MD available, a file system can be embedded into the **loader_4th** with */sys/tools/embed_mfs.sh*. Then, MD will be probed and be set to *currdev* during initialization.

Currently, MD is only supported in loader.efi(8).

FILES

/boot/loader	**loader_4th** itself.
/boot/boot.4th	Additional FICL initialization.
/boot/defaults/loader.conf	
/boot/loader.4th	Extra builtin-like words.
/boot/loader.conf	
/boot/loader.conf.local	**loader_4th** configuration files, as described in loader.conf(5).
/boot/loader.rc	**loader_4th** bootstrapping script.
/boot/loader.help	Loaded by **help**. Contains the help messages.
/boot/support.4th	*loader.conf* processing words.
/usr/share/examples/bootforth/	Assorted examples.

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_4th**:

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)

STANDARDS

For the purposes of ANS Forth compliance, loader is an *ANS Forth System with Environmental Restrictions, Providing* .(, :noname, ?do, parse, pick, roll, refill, to, value, \, false, true, <>, 0<>, compile, , erase, nip, tuck *and* marker *from the Core Extensions word set, Providing the Exception Extensions word set, Providing the Locals Extensions word set, Providing the Memory-Allocation Extensions word set, Providing* .s, bye, forget, see, words, [if], [else] *and* [then] *from the Programming-Tools extension word set, Providing the Search-Order extensions word set.*

HISTORY

The **loader_4th** first appeared in FreeBSD 3.1.

AUTHORS

The **loader_4th** was written by Michael Smith <msmith@FreeBSD.org>.

FICL was written by John Sadler <john_sadler@alum.mit.edu>.