SPARC OpenBoot and the Forth Language

by Boris Loza, PhD

From time to time as a UNIX system administrator, I've had to work in the Solaris Open Boot environment. It's useful for booting the operating system (*boot –r, boot cdrom –s,* etc.), modifying system start-up configuration parameters (*input-device, output-device, setenv,* etc.), troubleshooting (*probe-scsi-all, show-devs,* etc.), or running diagnostics (*test net, test /memory*, etc.). But sometimes, it isn't enough to use predefined commands and utilities. For this purpose, OpenBoot provides a very powerful environment based on the ANS Forth programming language.

Some Forth history

The name Forth was intended to suggest software for the fourth (next) generation computers, which Charles Moor (a programmer who invented it) saw as being characterized by distributed, small computers. The operating system he used at the time restricted filenames to five characters, so U was discarded. The first Forth interpreter was written in 1968. For the next five years, Forth was implemented on various CPUs and became widely known because of its high performance and economy of memory.

In 1988 Sun Microsystems invented Open Firmware technology—hardware-independent boot code, firmware, and device drivers. Open Firmware, then called OpenBoot, allows one version of the Boot ROM to run on any configuration of hardware and software. Such technology uses Forth as the official language.

Why Forth for all this?

Forth is a stack-based, extensible language without type-checking. It uses "reverse Polish" (postfix) arithmetic notation, familiar to users of HP calculators. To add two numbers in Forth, you'd type *2 3 +* instead of *2 + 3*. If you're using Forth, you don't need to recompile your program to add new functionality. You can define a new command and it will instantly be available for you to use. Because of this, the Forth compiler is simpler, smaller, and faster than other compilers. So the interactive Forth system, including an editor, assembler, and even multitasking support, can easily be put in an 8K EPROM!

The Forth language

The fundamental program unit in Forth is the *word*—a named data item, subroutine, or operator. Actually, OpenBoot commands such as `boot`, `printenv`, and `probe-scsi-all` are Forth-defined words. Programming in Forth consists of defining new words in terms of an existing one.

You can start programming in Forth at the OpenBoot *ok* prompt (*ok* is the usual prompt on Forth):

```
ok :  average (  a b – avg )  +  2/  .  ;
ok  10  20  average
ok  15
ok :  .ASCII  (  end start  --  ,  dump characters )
   do
     cr  i  .  i  emit  \  Print ASCII characters
   loop ;
ok  70  65  .ASCII
ABCDEF
ok
```

OpenBoot 3.x contains about 2,450 Forth words. All words belong to the dictionary that also contains vocabularies (consisting of related words and variables).

The new commands created above would be lost after rebooting a machine. OpenBoot provides a way to prevent this by saving into NVRAM using `nvedit`:

```
ok  nvedit
0:  :  hello  (  --  ) cr
1:   .” Welcome to OpenBoot!”  cr  ;
2:  ^C
ok  nvstore
ok  setenv  use-nvramrc?  true
ok  reset-all
--
ok  hello
Welcome to OpenBoot!
ok
```

By creating customized scripts, you can modify the OpenBoot start-up sequence. Unfortunately, you can't use the following commands here: `boot`, `go`, `nvedit`, `password`, `reset-all`, and `setenv security-mode`. OpenBoot provides various facilities for debugging Forth programs and loading and executing programs written in Forth from Ethernet, a hard disk, or a floppy device.

One of the Forth utilities that we'd like to mention here is the built-in Forth language decompiler—`see`. It can be used to re-create the source code for any previously defined Forth word. For instance:

```
ok  see scan-subtree
:  scan-subtree
[‘] scan-subtree  guarded-execute  drop
;
ok  see probe-scasi-all
(ffd60c5c)  [‘]  (ffd88b08)  scan-subtree
;
```

The preceding listing shows that `scan-subtree` is composed only of Forth source words that were compiled as *external* or as *headers* with `fcode-debug?` set to true. `probe-scsi-all` is a different word. It also contains words that were compiled as *hederless* and are, consequently, displayed as hex addresses surrounded by parentheses. For more information on how to use Forth development tools, consult the OpenBoot Reference Manual.

Forth and shell scripting

On the Internet, you can find various shareware ANS Forth compilers for a number of Operating Systems. One of the most interesting is *pForth* (a portable ANS stile Forth). After compiling and linking it on /usr/local/bin/forth, you can run standalone scripts like the one shown in **Listing A**. To run this program, just type the name of the file.

```
get-menu-item
```

Listing A:

A simple Forth script that can run on Solaris

```
#!/usr/local/bin/forth
\ \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
\   Enter a number:                                                    \
\   1.  Item number one                                                \
\   2.  Item number two                                                \
\   3.  Item number three                                              \
\                                                                      \
\   0.  Exit                                                           \
\ \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
: get-menu-item  ( -- n )  \  Query user for a menu option.
  begin
    ." Enter a number: "    cr
    ." 1. Item number one" cr
    ." 2. Item number two" cr
    ." 3. Item number three" cr  cr
    ." 0. Exit" cr
    key dup emit cr ascii 0 – dup 0 3 between not
  while
    drop
    ."  Not a valid menu number! Try again."  cr  cr
  repeat
case   ( case-value --  )
  1  of  ."  This is number one"       endof
  2  of  ."  This is number two"       endof
  3  of  ."  This is number three"     endof
  0  of  ."  Exit!" bye          endof
endcase  ;

get-menu-item
```

In our opinion, Forth cannot replace Perl and UNIX shell programming facilities for System Administration needs. These languages are specially designed for parsing strings and I/O manipulation. But you can practice with Forth scripting in order to be more comfortable with creating power tools in the OpenBoot environment.

References

1. "Scientific FORTH: a modern language for scientific computing" Julian V. Noble, Mechum Banks Publishing, 1992, 300 pages, ISBN: 0-9632775-0-2, disk included. (This book contains many serious examples of Forth programming style and useful programs.)
2. "Forth: The New Model—A Programmer's Handbook" Jack Woehr, M & T Publishing, 1992, 315 pages, ISBN: 0-13-036328-6, DOS disk included. (Describes features of ANS Forth and how to use it to write Forth programs. Currently the only book about ANS Forth.)
3. **www.forth.org/ – Forth Interest Group Home Page**
4. **http://pisa.rockefeller.edu:8080/FORTH/ - Forth on the Web**