Brief User's Guide for NoOSv2

January 2020
Dr. Robert Senser
rwsenser@gmail.com

Overview

NoOS is a PC software environment used to execute C programs without the support of a full operating system. The purpose of NoOS is to facilitate the execution of run-time performance tests without any additional activities running in parallel that can consume CPU cycles. A Linux system, the GNU C compiler suite and other software are required to build a new NoOS instance. The GRUB loader is used to load and start the NoOS instance. NoOS runs on PC-class hardware.

A NoOS instance can be used either on "bare-metal" hardware or in a virtual environment such as Virtual Box, VMware, Bochs, etc.

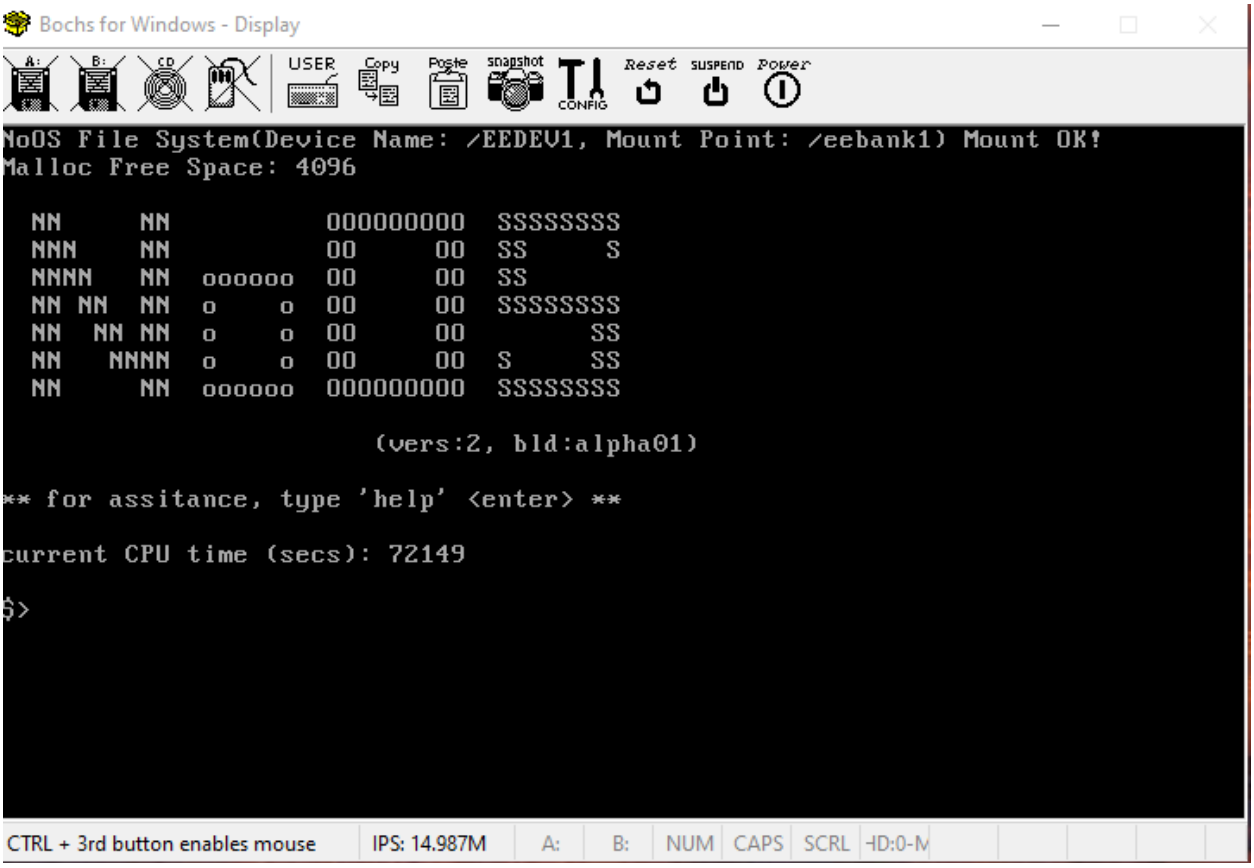

Figure 1.

Copyright and Licensing Considerations

Darnell Booklet Code

The source code supplied with the Darnell appears to be copyrighted and parts of it are used in NoOS. To avoid issues, please either purchase a copy of the Darnell Amazon Booklet for $2.99 (ASIN:

B01KU8N6FC) or obtain a pdf copy of the booklet, from the web. There may be a copy at: https://www.scribd.com/document/340290451/Create-Your-Own-Operating-System-Lucus-Darnell.

Source code from eefs

See https://github.com/nasa/eefs and https://opensource.gsfc.nasa.gov/projects/eefs/index.php. The eefs source used in NoOS has been modified to work with the NoOS "include" structure.

Senser-provided Code

The Senser-provided portions of NoOS are made available under the MIT license.

Demo Instance

The NoOS Git download comes with demo instances. After downloading, the contents of the "demos/baremetal" directory can be copied to a newly formatted thumb drive (or similar bootable medium). When a PC system is booted from this, NoOS will start. The resulting screen should look something like Figure 1.

Entering the "show-pgms" command will list the installed programs/apps. Entering "primes" will produce a list of prime numbers. Entering "help" will provide high-level help information and entering "exit" will cause NoOS to attempt to terminate or restart.

Running NoOS under Linux

It is also possible to run NoOS as a simple Linux batch program. This method of running NoOS does not show all the NoOS capabilities and it incurs the overhead and chaotic performance associated with using a full-function operating system. The "withOS" directory contains two linux scripts: "buildWos" and "run". The former will compile NoOS and latter executes it. "kernelWos" is the name of the created linux executable.

NoOS Installation & Build

To do a full installation and custom build of NoOS, with the capability to run new C code, the book by Darnell should be consulted. It provides installation instructions for the additional software needed to build a bare-metal version of NoOS. This booklet is available from Amazon (ASIN: B01KU8N6FC) and is also available from the web, as a pfd file.

Once all the required products, outlined in the Darnell booklet, have been installed on the build Linux system, the following NoOS steps should followed.

1) Download NoOSv2 from github at https://github.com/rwsenser/NoOSv2. Once the NoOSv2 directory is available, follow these steps in the director with "Makefile".
2) Enter " make clean".
3) Enter "make".
4) The newly created image is "noosv2". And iso version is available in "noosv2.iso".
5) The "demo" directory is NOT updated

The iso can be used in a virtual environment and the "noosv2" executable can be used on a bare-metal system. See the Darnell booklet for details or copy the needed support files from "demos/baremetal" and insert the new "noosv2".

Customization

To add new programs/apps to NoOS, the NoOS image must be rebuilt. See the “apps” directory and the “app_template.h” file. Reviewing the “prime.h” and “bobsort.h” files can show basic coding techniques. The “apps.h” and “appsnames.h” files need to be modified to link the new application with “kernel.c”.

Known Limits

1. Runs C programs and not C++ programs.
2. Run-time library is very, very constrained in terms of functionality, though it has more functionality than the basic Darnell version.
3. Run-time malloc() is (very) poorly implemented.
4. Limited RAM for program code.
5. Static stack size of 8,192 cannot be easily changed.
6. Limited in-core file system, based on eefs.
7. No support for Disk devices.
8. No support for USB drives (beyond GRUB booting).

Missing Features

1. IP support (should be possible – see the Darnell booklet).
2. Command line backspace support, etc.
3. Timers with better than 1 second resolution.

(see next page)

NoOS FAQs

1. Are there plans to future enhance NoOS? Not at this point, NoOS does what the original author intended.

2. How is NoOS different from the software provided with the Darnell booklet?
    a. Note: NoOS is built on the basics provided by Darnell.
    b. NoOS provides a file system.
    c. NoOS provides some limited timers (to the second).
    d. NoOS provides a stronger C-runtime library.
    e. NoOS provides an easy way to add new programs/apps (see “apps” directory).

3. How is NoOS licensed? The Senser-produced parts fall under the MIT license. The other components fall under their respective licenses (from example, see eefs software licensing).

4. Are there bugs in NoOS? It is extremely likely it has bugs. 😊 On the other hand, basic functionality has been tested. Many C-runtime API features are not supported. To see what is supported, look in the “NoOSrt” subdirectory and the associated “inc” and “src” subdirectories.

5. Why is the size of the generated image limited in size? NoOS is started by grub and it uses the default memory model supplied by grub. /* think of the DOS memory model */

6. Does NoOS run in 64-bit mode? No, it runs in 32-bit mode.

7. Why not just use freeDOS? NoOS does not start ANY background tasks and runs in 32-bit mode. Note however, there is likely a timer background interrupt occurring with NoOS.

8. Why are the timer values show by NoOS only to the second and not milli-second or nano-second? NoOS does not have any hardware-specific code and the simple timer capability in use is able to run on many (all??) PC platforms.

9. Why is the mode to run NoOS under Linux provided? This makes it easier to test and debug newly added applications. For final performance testing, the bare-metal version is used.

10. Why is the backspace key not supported in the kernel.c command line support? Every new features requires time for implementation and testing.

11. So what is a sample run like? Start NoOS and then enter “bobsort 25000” and then enter “bobsort 25000 -a”, and then notice the difference in run times. The “-a” sorts data already in ascending sequence.

12. I get a Bochs error at boot up when I try the demo, is that normal? Yes, just click “continue”; the Bochs config parameters for the included iso are not set correctly.