SEE TEXT ONLY

svn UPDATE

by Steven Kreuzer

In 2014, as a Google Summer of Code project, FreeBSD developers and students worked to design and implement a modular interface for the loader script interpreter—decoupling the interpreter from the loader. After four long years, it has finally been committed! FreeBSD 12 will ship with a brand-new bootloader based around Lua, which will finally make obsolete the faithful Forth loader that most say overstayed its welcome years ago. While most installments of this column tend to highlight development happening across all subsystems, I thought for this installment I would just focus on the bootloader, since I don't think I have ever seen so much activity happening on such a low-level component of the system.

Add Lua as a scripting language to /boot/loader— https://svnweb.freebsd.org/changeset/base/329166

liblua glues the lua runtime into the boot loader. It implements all the runtime routines that lua expects. In addition, it has a few standard 'C' headers that neuter various aspects of the LUA build that are too specific to lua to be in libsa. Many refinements from the original code improve implementation and the number of included lua libraries. Use int64_t for lua_Number. Have "/boot/lua" be the default module path. Numerous cleanups from the original GSoC project, including hacking libsa, allow lua to be built with only one change outside luaconf.h. Add the final bit of lua glue to bring in liblua, and plug into the multiple interpreter framework, previously committed.

Presently, this is an experimental option. One must opt-in to using this by defining WITH_LOADER_LUA and WITHOUT_FORTH. It's been lightly tested, so keep a backup copy of your old loader handy.

The menu code, coming in the next commit, hasn't been exhaustively tested. A LUA bootloader is 60k larger than a FORTH loader, which is 80k larger than a no-interpreter loader. Subtle changes in size may tip things past some subtle limit (the binary is ~430k now when built with LUA). A future version may offer coexistence.

Bump FreeBSD version to 1200058 to mark the milestone.

Add the Lua scripts from the Lua-bootloader SoC— https://svnweb.freebsd.org/changeset/base/329167

These are the .lua files from Pedro Souza's 2014 Summer of Code project. Rui Paulo, Pedro Arthur, and Wojciech A. Koszek also contributed.

Defer kernel/module loading until boot or menu escape— https://svnweb.freebsd.org/changeset/base/329576

Loading the kernel and modules can be really slow. Loading before the menu draws and every time one changes kernel/boot environment is even more painful. Defer loading until we either boot, auto-boot, or escape to loader prompt. We still need to deal with configuration changes as the boot environment changes, but this is generally much quicker. This commit strips all ELF loading out of config.load/config.reload so that these are purely for configuration. config.loadelf has been created to deal with kernel/module loads. Unloading logic has been ripped out, as we won't need to deal with it in the menu anymore.

Create a "carousel" menu entry type— https://svnweb.freebsd.org/changeset/base/329367

This is a precursor to boot environment support in lualoader. Create a new menu item type, "carousel_entry", that generally provides a callback to get the list of items, a carousel_id, for storing the current value and the standard name/func functions of an entry. The difference between this and a normal menu item, functionally, is that selecting a carousel item will automatically rotate through available items and wrap back at the beginning when the list is exhausted.

Re-work menu skipping bits— https://svnweb.freebsd.org/changeset/base/330020

This is motivated by a desire to reduce heap usage if the menu is being skipped. Currently, the menu module must be loaded regardless of whether it is being skipped or not, which adds a cool ~50-100KB worth of memory usage. Move the menu skip logic out to core (and remove a debug print), then check in loader.lua if we should be skipping the menu and avoid loading the menu module entirely. This keeps our memory usage below ~115KB for a boot with the menu stripped.

Fix module_path handling with multiple kernels— https://svnweb.freebsd.org/changeset/base/329497

Once we've successfully loaded a kernel, we add its directory to module_path. If we switch kernels with the kernel selector, we again prepend the kernel directory to the current module_path and end up with multiple kernel paths, potentially with mismatched kernel/modules added to module_path. Fix it by caching module_path at load() time and using the cached version whenever we load a new kernel.

Invalidate the screen from menu perspective upon menu exits— https://svnweb.freebsd.org/changeset/base/329986

In the common case, this will effectively do nothing as the menu will get redrawn as we leave submenus, regardless of whether the screen has been marked invalid or not. However, upon escape to the loader prompt, one could do either of the following to reenter the menu system:

— Method 1

```
require('menu').run()
```

— Method 2

```
require('menu').process(menu.default)
```

With method 1, the menu will get redrawn anyway, as we do this before autoboot checking upon entry. With method 2, however, the menu will not be redrawn without this invalidation.

Both methods are acceptable for reentering the menu system, although the latter method in the local module for processing new and interesting menus is more expected.

STEVEN KREUZER is a FreeBSD Developer and Unix Systems Administrator with an interest in retro-computing and air-cooled Volkswagens. He lives in Queens, New York, with his wife, daughter, and dog.