NetBSD Google Summer of Code 2024 Project Proposal

puffs(3) bindings for Lua + SquashFS in Lua

Author: James Cave
GitHub: https://github.com/jtcave
Email: jcave137@gmail.com

About the project

NetBSD has several unique features that set it apart from other operating systems. These features, like most low-level things in Unix, have mostly been written to be accessed from C code. It would be convenient to access these features from a high level language. NetBSD happens to include the Lua language in its base system, and already exposes some facilities, such as `gpio(3lua)`. Lua could become a good fit for NetBSD.

To that end, this project has two goals.

First, I will write a Lua module that exposes the puffs interface to Lua scripts. This will let people write filesystems for NetBSD in a high level language. The module's API will be designed with Lua's features in mind. In particular, the single-threaded cooperative multitasking that puffs uses can be achieved naturally with Lua coroutines.

Second, as a practical demonstration of the module, I will write an implementation of SquashFS, a compressed read-only filesystem, and create a `mount_squashfs(8)` command. To a user, it will behave like any other filesystem; they will not need to know that it is a userspace filesystem written in Lua.

Deliverables

The following deliverables will be made:

- A Lua module, puffs.so, that exposes the `puffs(3)` interface to Lua scripts
- An implementation of SquashFS, a read-only compressed filesystem

    - This will require creating a `libz(3lua)` binding
- Demos:
    - A filesystem written in Lua that stores files in an `sqlite(3lua)` database
    - Store /emul/linux in a single SquashFS image (allows easily upgrading or swapping the Linux userland)
- Man pages for all commands and libraries produced.
- A writeup describing the new module, demonstrations, and development process.

Challenges and risks

I have identified some challenges that will need to be overcome for successful completion of the project:

- Very few NetBSD-specific APIs are accessible from Lua. While I do not foresee needing to access any libraries other than zlib, the project could end up requiring an unforeseen system facility or other library. Having to expose more functionality to Lua could jeopardize my ability to maintain the schedule.

- Lua coroutines and puffs continuations both implement cooperative multitasking, but they are implemented incompatibly. Lua uses `longjmp(3)`, while puffs uses `swapcontext(3)`. It is likely that the puffs continuation framework will have to be replaced with a Lua-specific coroutine scheduler and event loop.

- This binding will create filesystems that use coroutines to handle concurrent operations on a single thread. Designing a pleasant high-level API for this will require some thought.

- As shipped in NetBSD, Lua does not have asynchronous I/O. The third-party libraries that implement it do not seem to be compatible with each other. I will also have to write the zlib binding to whichever concurrency model I use, which may make it less useful to other consumers.

Proposed schedule

This will be a “medium” size project (175 hours).

Week of	*Task*	*Deliverable*
May 1-26	*Community bonding period*. Design Lua-friendly API for the puffs and zlib bindings. Further study of SquashFS spec. Figure out how to test Lua scripts with ATF. Figure out how coroutine concurrency is going to be exposed.	
May 27	**Coding starts**. Begin writing puffs binding. Mount a “hello world” filesystem. (This will require partial bindings to the `puffs_node(3)` and `puffs_ops(3)` subsystems.)	
June 3	Write the coroutine event loop.	
June 10	Fill out remaining _node and _ops routines.	
June 17	Implement _flush, _creds, and _path.	
June 24	Write unit tests and documentation.	
July 1	Write the sqlitefs demo.	`sqlitefs` demo
July 8	**Midterm** - evals, writeup, deliver binding	`puffs(3lua)`
July 15	Write or integrate zlib binding. Write Lua functions to dissect SquashFS images.	`zlib(3lua)`
July 22	Integrate SquashFS dissection routines with the puffs binding to obtain basic functionality	
July 29	Obtain and create SquashFS filesystems to stress test the implementation. Write automated tests. Fix bugs. Test on ppc and sparc.	
Aug 5	Performance testing with various benchmarks.	`mount_squashfs(8)`
Aug 12	Experiment with storing the contents of /emul/linux in SquashFS for ease of distribution	`squashemul` demo
Aug 19	**Final week** - writeup on the whole endeavor, finish documentation	Writeup

Possible tasks for a project extension include:

- support for more compression algorithms in SquashFS
- general performance tuning
- an implementation of another filesystem
- refactoring the coroutine dispatch loop into a separate module

About the project and NetBSD

Experience with NetBSD

I am familiar with the process for installing NetBSD using `sysinst(8)`, as well as the process to upgrade the system from source as listed in the NetBSD Guide. I have built packages from pkgsrc, but I have yet to create my own packages.

I have read the documentation for the puffs API, as well as some of the sample programs and test cases in the source tree. I have also learned the Lua language, and I have practiced using the C API to create an extension module.

SquashFS information

Documentation for the SquashFS filesystem appears to be available at https://dr-emann.github.io/squashfs/squashfs.html. To allow the release of this code under a BSD license, I will be unable to use Linux kernel code or `squashfs-tools` code as a source. (But I can still use the squashfs-tools themselves and reverse engineer the output.)

Related work

FUSE bindings for Lua already exist. The most downloaded one listed on LuaRocks is dromozoa-fuse. These could be easily adapted to bind to the `refuse(3)` library. But I think the coroutine-based concurrency model that puffs uses is a more natural fit for Lua than the thread-based concurrency of FUSE.

There exists squashfuse, a FUSE implementation of SquashFS (in C) that claims to be compatible with NetBSD. This could be used in performance testing to measure the performance impact of using Lua.

LuaJIT has an FFI library that generates bindings from C declarations. This would allow for rudimentary bindings to be written rapidly. But LuaJIT is not available for all architectures NetBSD supports, nor is the (unmaintained) port to regular Lua. Creating a C module is the only way to have a chance to support all machines.

About me

I've been following NetBSD for a very long time, ever since discovering the SDF system in the early 2000s. Watching NetBSD evolve and innovate over the years while staying true to its portable Unix roots has been wonderful. I would be honored to contribute to it.

I don't have any experience with developing filesystems, though I do know about some of the underlying concepts (VFS, inodes, vnodes, etc.). In preparation for this project, I've read the papers on puffs mentioned on the NetBSD documentation site, as well as relevant man pages like `puffs(3)` and `vnode(9)`.

As far as personal projects go, I have written a crude Scheme interpreter, posted at https://github.com/jtcave/lilscheme. While it isn't even close to completion, it should be a representative sample of my C skills.

Thank you for reading my project proposal. If you have any feedback or want to get in touch, feel free to send a reply to the tech-userlevel mailing list, or email me directly at jcave137@gmail.com.

~ James Cave