FreeBSD at 30 Years: Its Secrets to Success

BY MARSHALL KIRK MCKUSICK

This year the FreeBSD Project is celebrating its thirtieth year of providing a complete system distribution. The goal of this article is to understand what it is that has made FreeBSD one of the few long-term, viable, open-source projects. Most projects with long-term successes are sponsored by companies that base their products around the open-source software that they actively nurture. While FreeBSD has companies actively using and supporting it, they have come and gone over the years and no single company has been the primary long-term proponent.

Origin

Many open-source projects start with code written by one person and begin building from there. FreeBSD started from a solid code base, the 4.4BSD-Lite open-source distribution from the University of California at Berkeley. The Berkeley Software Distribution (BSD) had been in development and distribution for over a decade and the BSD distribution started from the Unix distribution from Bell Laboratories that had been in development for a decade before BSD. Though BSD was not open source, its code was widely licensed and had many contributors from both academia and industry. Nearly all of BSD was ultimately released as open source in the 4.4BSD-Lite distribution.

When the FreeBSD organization was set up, the organizers decided to establish a group of seven people called the Core group that oversaw the project.

The BSD kernel introduced important operating-system interfaces still used today:

- the socket networking interface and the original and widely used implementation of TCP/IP,
- the set of system calls used to operate on filesystems, the virtual filesystem (VFS) interface to support multiple filesystem implementations, and the fast file system and network filesystem (NFS) implementations,
- the mmap memory model, and
- the interface to manage processes (signals, process groups, job control, etc.)

The BSD distributions also established the model of complete system distributions that included the operating system, a core set of libraries and utilities, contributed software (that would eventually become FreeBSD's ports), and complete manual pages and system documentation.

Leadership

Most open-source projects are started by a single person who then becomes the czar-for-life leader of the project. A well-known example is Linus Torvalds who created and still leads the Linux project. Projects usually go dark when the leader loses interest and stops working on it. Contributors often get frustrated if the leader is not good at reviewing and critiquing or accepting input from others.

When the FreeBSD organization was set up, the organizers decided to establish a group of seven people called the Core group that oversaw the project. The original Core group was self-selected. The people who set up the project deputized themselves onto the Core team. They were "Czars for life." The Core team decides project direction and awards and removes the privilege of being a committer; committers are the people who are allowed to make changes to the project repository.

While this approach was better than having a single leader, it still had the problem that committers could only rise to a middle level in the project, thus leading to frustration and abandonment if their ideas were not accepted. To remedy this, the FreeBSD project decided to make Core an elected position. Core was also expanded to nine people. The entire Core is elected every two years. Core members are nominated from and elected by the committers. Any active committer can run for Core. Candidates are self-selecting and no nomination is required. The effect of this change is that newcomers can rise to leadership roles. As a result, the project leadership evolves over time, and the project is much less susceptible to collapse if its leader departs.

Development

From its inception, the FreeBSD project used centrally located tools (source-code control and bug reporting). This tooling enabled remote development from the start. Though common today, at the time FreeBSD was started, the usual approach was to have a single person who maintained the distribution, and changes by others had to be sent to them for inclusion. As the project grew, the person maintaining the master copy of the source would get overloaded and limit the speed with which the project could move forward. It also made it difficult to keep track of who was working on what when bugs would arise and needed to be assigned. Happily, the modern tool sets available today like gitlab and github mitigate these issues.

The FreeBSD project has also benefited greatly from adopting ideas and code from the NetBSD and OpenBSD projects. NetBSD has lead the way in efficiently supporting multiple architectures which was very helpful as FreeBSD began expanding from its

initial focus on the Intel architecture to support additional architectures. NetBSD also has provided many tests that have been incorporated into the FreeBSD continuous-integration testing. OpenBSD has focused on system security and FreeBSD has incorporated many of their security improvements. OpenBSD has also provided several of the key security components used in FreeBSD such as the ssh remote access and login program and the software components that support https encryption.

Distributions

Many open-source projects are simply a collection of code that must be downloaded, compiled, and installed to be used. They often depend on other libraries and infrastructure which must also be found, built, and installed. In recent years, projects are beginning to provide containers that can be spun up, though they are an inefficient use of resources since they include the entire software stack all the way down to and often including the operating system, thus duplicating vast amounts of software already on the machine.

Early in the FreeBSD project history it began distributing CD-ROMs with the complete system on them that could be booted on PC computers. Users could boot up the system from the CD-ROM to try it out and then install it on their hard disk if they wished to do so. And--since it was derived from the BSD system from which it started--all the commands and libraries that they needed were already there. Prolific documentation was provided, making installation easy even for non-experts.

Hardware Support

Most open-source projects try to support everything, which usually means much hardware performs poorly and often fails under load. From the start of the FreeBSD project, the decision was made to curate hardware and decide what worked well with FreeBSD. Once the hardware was selected, significant effort was made to write robust and complete device drivers to run it. FreeBSD published a list of hardware that they recommended and supported that hardware by fixing reported problems and updating drivers as newer versions of the hardware were released. This curated list made it easy to put together server machines that ran well under load. FreeBSD became the system of choice for companies running dial-up servers and later Internet and web server providers because they had great performance and ran reliably.

Communication

Since nearly all the FreeBSD developers were working remotely, it was important to set up mailing lists to discuss core design decisions. Topic areas included networking, filesystems, core architecture, etc. A frequent issue with mailing lists, especially when most folks on them have never met, is that discussion can get off-track and distinctly nasty. Flamewars were not uncommon in the first few years of the project, so the mailing lists began to be actively monitored to tamp down bad behavior and ensure civil discussion. Sadly, many projects even today have toxic mailing lists. Once a project gets a reputation for bad behavior, it often results in it entering a death spiral. Alternatively, it is possible to go to the opposite extreme and become so controlling that folks abandon the project as they feel overly constrained. And for projects like FreeBSD that have developers worldwide, it can be difficult to find rules that work in the large diversity of cultures of its developers. The problem is never solved; ultimately there needs to be an ever-evolving methodology on how to keep the project moving forward on an even keel.

Documentation

The FreeBSD project started off with a solid base of documentation based on the documentation in the 4.4BSD-Lite distribution which was in turn derived from documentation in the UNIX system from which BSD evolved. Early in its evolution, FreeBSD embraced contributors that focused on system documentation. Folks writing code were encouraged to work with those writing the documentation to ensure that the documentation was complete and correct.

The project set up a documentation committer group for the folks doing the documentation. This group was given all the rights and privileges of code committers. They could run for Core, had equal voting rights, and their own group leaders that handled adding and removing documentation committers, setting up the documentation structure and tools, and overseeing the document repository. Under their direction the documentation was structured with a framework that allowed it to easily support multiple languages. Many of the documentation committers started out by doing translations of documents into their native language. This translation task often helped them get up to speed both on how the documentation tools worked and how FreeBSD itself worked.

Since nearly all the FreeBSD developers were working remotely, it was important to set up mailing lists to discuss core design decisions.

The Ports Collection

The 4.4BSD-Lite distribution had a collection of contributed software that consisted of about fifty utilities and libraries that had been developed outside Berkeley but were included in the BSD distributions. These included things like the X window system, the gated routing daemon, the emacs editor, etc. FreeBSD started with this set of core contributed programs and greatly expanded on it with what became the ports collection. Unlike the BSD distribution which installed all the contributed programs, FreeBSD ports provided them separately so that individual sites could install only those that they needed. The ports collection ensured that the program would compile and run on FreeBSD with reasonable defaults. It also ensured that fixes for bugs found in the BSD environment were up streamed to the maintainer of the

software and that changes made up stream were brought down to the FreeBSD port. Most users could just use the compiled version of the port though those needing site-specific changes could make them and then build their own binaries. The port collection made it easy to use other open-source software on FreeBSD. Having a ports equivalent is done by most open-source distributors today but was new at the time.

The ports collection has continued to evolve over the years. Recent innovations are the addition of pkg system to manipulate ports. The pkg system handles registering, adding, removing, and upgrading packages. The other key component is Poudriere that is a utility for creating and testing FreeBSD packages. It uses FreeBSD jails to set up isolated compilation environments. These jails can be used to build packages for versions of FreeBSD that are different from the system on which it is installed and to build packages for a different architecture than the host system. Once the packages are built, they are in a layout identical to the official mirrors. These packages are usable by the pkg system and other package management tools.

FreeBSD provides a base platform that can be modified to build a customized OS along with all the infrastructure needed to build a full OS distribution including not just the base system but also a collection of the ports. The OS can be customized to support an appliance as all the bits for how to build the release image for the customized OS along with automated building of packages via Poudriere for the customized OS are public and well-documented. None of the Linux distributions are as turnkey as FreeBSD in this regard. For example, it would be much more difficult to build your own Debian-fork on top of a modified kernel and system libraries, etc.

> Port, documentation, and development committers are all given equal say in how the project is run.

Project Culture

Port, documentation, and development committers are all given equal say in how the project is run. Notably, they all can run for Core and get the same voting rights. In most projects, the developers have more say and others are treated as inferior. The FreeBSD project has worked on building a culture of inclusion from its start. The culture values ''plays well with others'' above anything else. It does not tolerate a diva just to get their documentation, port, or code (though sometimes it can take a while to get to the point of a diva leaving or getting kicked out).
The FreeBSD project is not set up to train people how to write or program. Folks joining the FreeBSD project are expected to know their trade. Documentation writers are expected to know how to write technical documents. Port and source contributors are expected to know C and any other relevant languages along with the tools used to write, build, debug, and profile them. That said, FreeBSD has been involved with mentoring students through programs such as Google's Summer of Code. Indeed, many of the students in Summer of Code have gone on to become committers on the FreeBSD Project.

The FreeBSD project is welcoming to new folks. It is not necessary to survive a gauntlet of hazing or needing to ingratiate yourself to the project leader to become a project committer. There is a well-documented process on how to become involved with the project.

Project Support

When FreeBSD started, its infrastructure was a machine in a developer's home. As it grew, its infrastructure was supported first by Walnut Creek CD-ROM and later by Yahoo. Being dependent on a company's goodwill was a recipe for disaster, so the FreeBSD Foundation was created to raise money whose initial use was to provide the machines and hosting for FreeBSD infrastructure. While Foundation support for projects is common today, FreeBSD was one of the first projects to set up a foundation to support the project. The Foundation was originally run by its (unpaid) board of directors. After a few years, it was able to hire its first part-time employee. Today it has nearly twenty staff and contractors supporting infrastructure, development, marketing, tooling, fund raising, and other project-related services.

Licensing

FreeBSD uses a Berkeley license which does not require companies to make their code available to others. The use of the Berkeley license has played a big role in FreeBSD's success, particularly with companies that have their proprietary code in the kernel. FreeBSD is heavily used in the appliance and embedded operating system market where companies need to put their intellectual property inside the operating system and thus cannot use Linux due to its GNU Public License (GPL) that requires source code for all changes be made available.

Conclusions

FreeBSD is still going strong. Its strength comes from having built a strong base in its code, documentation, and culture. It has managed to evolve with the times, continuing to bring in new committers, and smoothly transition through several leadership groups. It continues to fill an important area of support that is an alternative to Linux. Specifically, companies needing redundancy require more than one operating system, since any single operating system may fall victim to a failure that could take out the entire company's infrastructure. For all these reasons, FreeBSD has a bright future. In short, FreeBSD is awesome!

DR. MARSHALL KIRK MCKUSICK writes books and articles, teaches classes on UNIX- and BSD-related subjects, and provides expert-witness testimony on software patent, trade secret, and copyright issues. He has been a developer and committer to the FreeBSD Project since its founding in 1993.