Building Products with FreeBSD

Abstract

The FreeBSD project is a worldwide, volunteer based, and collaborative project, which develops a portable and high-quality operating system. The FreeBSD project distributes the source code for its product under a liberal license, with the intention of encouraging the use of its code. Collaborating with the FreeBSD project can help organizations reduce their time to market, reduce engineering costs and improve their product quality.

This article examines the issues in using FreeBSD code in appliances and software products. It highlights the characteristics of FreeBSD that make it an excellent substrate for product development. The article concludes by suggesting a few "best practices" for organizations collaborating with the FreeBSD project.

Table of Contents

1. Introduction . . . 1
2. FreeBSD as a set of building blocks . . . 2
3. Collaborating with FreeBSD . . . 6
4. Conclusion . . . 9
Bibliography . . . 10

1. Introduction

FreeBSD today is well-known as a high-performance server operating system. It is deployed on millions of web servers and internet-facing hosts worldwide. FreeBSD code also forms an integral part of many products, ranging from appliances such as network routers, firewalls, and storage devices, to personal computers. Portions of FreeBSD have also been used in commercial shrink-wrapped software (see FreeBSD as a set of building blocks).

In this article we look at the FreeBSD project as a software engineering resource-as a collection of building blocks and processes which you can use to build products.

While FreeBSD's source is distributed freely to the public, to fully enjoy the benefits of the project's work, organizations need to *collaborate* with the project. In subsequent sections of this article we discuss effective means of collaboration with the project and the pitfalls that need to be avoided while doing so.

Caveat Reader. The author believes that the characteristics of the FreeBSD Project listed in this article were substantially true at the time the article was conceived and written (2005). However, the reader should keep in mind that the practices and processes used by open-source communities can change over time, and that the information in this article should therefore be taken as

indicative rather than normative.

1.1. Target Audience

This document would be of interest to the following broad groups of people:

- Decision makers in product companies looking at ways to improve their product quality, reduce their time to market and lower engineering costs in the long term.
- Technology consultants looking for best-practices in leveraging "open-source".
- Industry observers interested in understanding the dynamics of open-source projects.
- Software developers seeking to use FreeBSD and looking for ways to contribute back.

1.2. Article Goals

After reading this article you should have:

- An understanding of the goals of the FreeBSD Project and its organizational structure.
- An understanding of its development model and release engineering processes.
- An understanding of how conventional corporate software development processes differ from that used in the FreeBSD project.
- Awareness of the communication channels used by the project and the level of transparency you can expect.
- Awareness of optimal ways of working with the project-how best to reduce engineering costs, improve time to market, manage security vulnerabilities, and preserve future compatibility with your product as the FreeBSD project evolves.

1.3. Article Structure

The rest of the article is structured as follows:

- FreeBSD as a set of building blocks introduces the FreeBSD project, explores its organizational structure, key technologies and release engineering processes.
- Collaborating with FreeBSD describes ways to collaborate with the FreeBSD project. It examines common pitfalls encountered by corporates working with voluntary projects like FreeBSD.
- Conclusion concludes.

2. FreeBSD as a set of building blocks

FreeBSD makes an excellent foundation on which to build products:

- FreeBSD source code is distributed under a liberal BSD license facilitating its adoption in commercial products Why you should use a BSD style license for your Open Source Project with minimum hassle.

- The FreeBSD project has excellent engineering practices that can be leveraged.
- The project offers exceptional transparency into its workings, allowing organizations using its code to plan effectively for the future.
- The culture of the FreeBSD project, carried over from the Computer Science Research Group at The University of California, Berkeley Twenty Years of Berkeley Unix: From AT&T-Owned to Freely Redistributable, fosters high-quality work. Some features in FreeBSD define the state of the art.

Innovation Happens Elsewhere: Open Source as Business Strategy examines the business reasons for using open-source in greater detail. For organizations, the benefits of using FreeBSD components in their products include a shorter time to market, lower development costs and lower development risks.

2.1. Building with FreeBSD

Here are a few ways organizations have used FreeBSD:

- As an upstream source for tested code for libraries and utilities.

  By being "downstream" of the project, organizations leverage the new features, bug fixes and testing that the upstream code receives.

- As an embedded OS (for example, for an OEM router and firewall device). In this model, organizations use a customized FreeBSD kernel and application program set along with a proprietary management layer for their device. OEMs benefit from new hardware support being added by the FreeBSD project upstream, and from the testing that the base system receives.

  FreeBSD ships with a self-hosting development environment that allows easy creation of such configurations.

- As a Unix compatible environment for the management functions of high-end storage and networking devices, running on a separate processor "blade".

  FreeBSD provides the tools for creating dedicated OS and application program images. Its implementation of a BSD unix API is mature and tested. FreeBSD can also provide a stable cross-development environment for the other components of the high-end device.

- As a vehicle to get widespread testing and support from a worldwide team of developers for non-critical "intellectual property".

  In this model, organizations contribute useful infrastructural frameworks to the FreeBSD project (for example, see netgraph(3)). The widespread exposure that the code gets helps to quickly identify performance issues and bugs. The involvement of top-notch developers also leads to useful extensions to the infrastructure that the contributing organization also benefits from.

- As a development environment supporting cross-development for embedded OSes like RTEMS and eCOS.

There are many full-fledged development environments in the 37000-strong collection of applications ported and packaged with FreeBSD.

- As a way to support a Unix-like API in an otherwise proprietary OS, increasing its palatability for application developers.

  Here parts of FreeBSD's kernel and application programs are "ported" to run alongside other tasks in the proprietary OS. The availability of a stable and well tested Unix™ API implementation can reduce the effort needed to port popular applications to the proprietary OS. As FreeBSD ships with high-quality documentation for its internals and has effective vulnerability management and release engineering processes, the costs of keeping up-to-date are kept low.

2.2. Technologies

There are a large number of technologies supported by the FreeBSD project. A selection of these are listed below:

- A complete system that can cross-host itself for many architectures:
- A modular symmetric multiprocessing capable kernel, with loadable kernel modules and a flexible and easy to use configuration system.
- Support for emulation of Linux™ and SVR4 binaries at near machine speeds. Support for binary Windows™ (NDIS) network drivers.
- Libraries for many programming tasks: archivers, FTP and HTTP support, thread support, in addition to a full POSIX™ like programming environment.
- Security features: Mandatory Access Control (mac(9)), jails (jail(2)), ACLs, and in-kernel cryptographic device support.
- Networking features: firewall-ing, QoS management, high-performance TCP/IP networking with support for many extensions.

  FreeBSD's in-kernel Netgraph (netgraph(4)) framework allows kernel networking modules to be connected together in flexible ways.

- Support for storage technologies: Fibre Channel, SCSI, software and hardware RAID, ATA and SATA.

  FreeBSD supports a number of filesystems, and its native UFS2 filesystem supports soft updates, snapshots and very large filesystem sizes (16TB per filesystem) Soft Updates: A Technique for Eliminating Most Synchronous Writes in the Fast Filesystem.

  FreeBSD's in-kernel GEOM (geom(4)) framework allows kernel storage modules to be composed in flexible ways.

- Over 37000 ported applications, both commercial and open-source, managed via the FreeBSD ports collection.

2.3. Organizational Structure

FreeBSD's organizational structure is non-hierarchical.

There are essentially two kinds of contributors to FreeBSD, general users of FreeBSD, and developers with write access (known as *committers* in the jargon) to the source base.

There are many thousands of contributors in the first group; the vast majority of contributions to FreeBSD come from individuals in this group. Commit rights (write access) to the repository are granted to individuals who contribute consistently to the project. Commit rights come with additional responsibilities, and new committers are assigned mentors to help them learn the ropes.

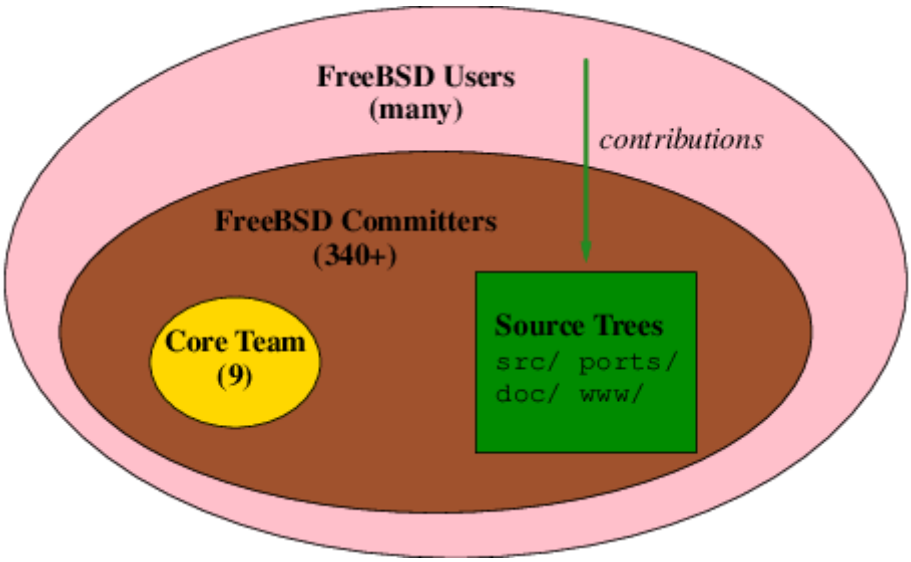


Figure 1. FreeBSD Organization

Conflict resolution is performed by a nine member "Core Team" that is elected from the group of committers.

FreeBSD does not have "corporate" committers. Individual committers are required to take responsibility for the changes they introduce to the code. The FreeBSD Committer's guide Committer's Guide documents the rules and responsibilities for committers.

FreeBSD's project model is examined in detail in A project model for the FreeBSD Project.

2.4. FreeBSD Release Engineering Processes

FreeBSD's release engineering processes play a major role in ensuring that its released versions are of a high quality. At any point of time, FreeBSD's volunteers support multiple code lines (FreeBSD Release Branches):

- New features and disruptive code enters on the development branch, also known as the *-CURRENT* branch.
- *-STABLE* branches are code lines that are branched from HEAD at regular intervals. Only tested code is allowed onto a -STABLE branch. New features are allowed once they have been tested and stabilized in the -CURRENT branch.
- *-RELEASE* branches are maintained by the FreeBSD security team. Only bug fixes for critical

issues are permitted onto -RELEASE branches.

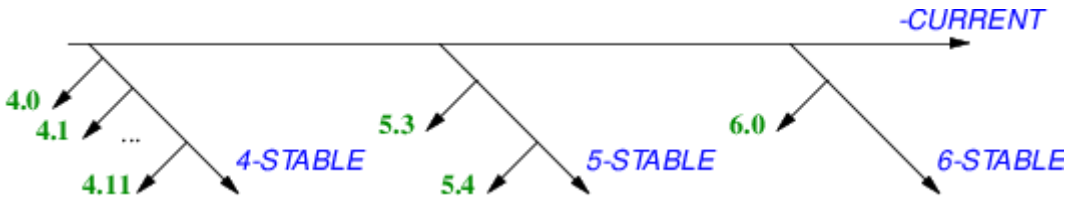


Figure 2. FreeBSD Release Branches

Code lines are kept alive for as long as there is user and developer interest in them.

Machine architectures are grouped into "tiers"; *Tier 1* architectures are fully supported by the project's release engineering and security teams, *Tier 2* architectures are supported on a best effort basis, and experimental architectures comprise *Tier 3*. The list of supported architectures is part of the FreeBSD documentation collection.

The release engineering team publishes a road map for future releases of FreeBSD on the project's web site. The dates laid down in the road map are not deadlines; FreeBSD is released when its code and documentation are ready.

FreeBSD's release engineering processes are described in FreeBSD Release Engineering.

3. Collaborating with FreeBSD

Open-source projects like FreeBSD offer finished code of a very high quality.

While access to quality source code can reduce the cost of initial development, in the long-term the costs of managing change begin to dominate. As computing environments change over the years and new security vulnerabilities are discovered, your product too needs to change and adapt. Using open-source code is best viewed not as a one-off activity, but as an *ongoing process*. The best projects to collaborate with are the ones that are *live*; i.e., with an active community, clear goals and a transparent working style.

- FreeBSD has an active developer community around it. At the time of writing there are many thousands of contributors from every populated continent in the world and over 300 individuals with write access to the project's source repositories.
- The goals of the FreeBSD project are Contributing to the FreeBSD Project:
  - To develop a high-quality operating system for popular computer hardware, and,
  - To make our work available to all under a liberal license.
- FreeBSD enjoys an open and transparent working culture. Nearly all discussion in the project happens by email, on public mailing lists that are also archived for posterity. The project's policies are documented and maintained under revision control. Participation in the project is open to all.

3.1. Understanding FreeBSD culture

To be able to work effectively with the FreeBSD project, you need to understand the project's

culture.

Volunteer driven projects operate under different rules than for-profit corporates. A common mistake that companies make when venturing into the open-source world is that of underplaying these differences.

Motivation. Most contributions to FreeBSD are done voluntarily without monetary rewards entering the picture. The factors that motivate individuals are complex, ranging from altruism, to an interest in solving the kinds of problems that FreeBSD attempts to solve. In this environment, "elegance is never optional"Tutorial on Good Lisp Programming Style.

The Long Term View. FreeBSD traces its roots back nearly twenty years to the work of the Computer Science Research Group at the University of California Berkeley.[1] A number of the original CSRG developers remain associated with the project.

The project values long-term perspectives Teach Yourself Programming in Ten Years. A frequent acronym encountered in the project is DTRT, which stands for "Do The Right Thing".

Development Processes. Computer programs are tools for communication: at one level programmers communicate their intentions using a precise notation to a tool (a compiler) that translates their instructions to executable code. At another level, the same notation is used for communication of intent between two programmers.

Formal specifications and design documents are seldom used in the project. Clear and well-written code and well-written change logs (A sample change log entry) are used in their place. FreeBSD development happens by "rough consensus and running code"The Architectural Principles of the Internet.

```
r151864 | bde | 2005-10-29 09:34:50 -0700 (Sat, 29 Oct 2005) | 13 lines
Changed paths:
   M /head/lib/msun/src/e_rem_pio2f.c

Use double precision to simplify and optimize arg reduction for small
and medium size args too: instead of conditionally subtracting a float
17+24, 17+17+24 or 17+17+17+24 bit approximation to pi/2, always
subtract a double 33+53 bit one.  The float version is now closer to
the double version than to old versions of itself -- it uses the same
33+53 bit approximation as the simplest cases in the double version,
and where the float version had to switch to the slow general case at
|x| == 2^7*pi/2, it now switches at |x| == 2^19*pi/2 the same as the
double version.

This speeds up arg reduction by a factor of 2 for |x| between 3*pi/4 and
2^7*pi/4, and by a factor of 7 for |x| between 2^7*pi/4 and 2^19*pi/4.
```

A sample change log entry

Communication between programmers is enhanced by the use of a common coding standard style(9).

Communication Channels. FreeBSD's contributors are spread across the world. Email (and to a lesser extent, IRC) is the preferred means of communication in the project.

3.2. Best Practices for collaborating with the FreeBSD project

We now look at a few best practices for making the best use of FreeBSD in product development.

Plan for the long term

Setup processes that help in tracking the development of FreeBSD. For example:

Track FreeBSD source code. The project makes it easy to mirror its SVN repository using svnsync. Having the complete history of the source is useful when debugging complex problems and offers valuable insight into the intentions of the original developers. Use a capable source control system that allows you to easily merge changes between the upstream FreeBSD code base and your own in-house code.

An annotated source listing generated using `svn blame` shows a portion of an annotated listing of the file referenced by the change log in A sample change log entry. The ancestry of each line of the source is clearly visible. Annotated listings showing the history of every file that is part of FreeBSD are available on the web.

```
#REV         #WHO #DATE                                       #TEXT

176410        bde 2008-02-19 07:42:46 -0800 (Tue, 19 Feb 2008) #include
<sys/cdefs.h>
176410        bde 2008-02-19 07:42:46 -0800 (Tue, 19 Feb 2008)
__FBSDID("$FreeBSD$");
  2116        jkh 1994-08-19 02:40:01 -0700 (Fri, 19 Aug 1994)
  2116        jkh 1994-08-19 02:40:01 -0700 (Fri, 19 Aug 1994) /*
__ieee754_rem_pio2f(x,y)
  8870    rgrimes 1995-05-29 22:51:47 -0700 (Mon, 29 May 1995)  *
176552        bde 2008-02-25 05:33:20 -0800 (Mon, 25 Feb 2008)  * return the
remainder of x rem pi/2 in *y
176552        bde 2008-02-25 05:33:20 -0800 (Mon, 25 Feb 2008)  * use double
precision for everything except passing x
152535        bde 2005-11-16 18:20:04 -0800 (Wed, 16 Nov 2005)  * use
__kernel_rem_pio2() for large x
  2116        jkh 1994-08-19 02:40:01 -0700 (Fri, 19 Aug 1994)  */
  2116        jkh 1994-08-19 02:40:01 -0700 (Fri, 19 Aug 1994)
176465        bde 2008-02-22 07:55:14 -0800 (Fri, 22 Feb 2008) #include <float.h>
176465        bde 2008-02-22 07:55:14 -0800 (Fri, 22 Feb 2008)
  2116        jkh 1994-08-19 02:40:01 -0700 (Fri, 19 Aug 1994) #include "math.h"
```

An annotated source listing generated using `svn blame`

Use a gatekeeper. Appoint a *gatekeeper* to monitor FreeBSD development, to keep an eye out for changes that could potentially impact your products.

Report bugs upstream. If you notice bug in the FreeBSD code that you are using, file a bug report. This step helps ensure that you do not have to fix the bug the next time you take a code drop from upstream.

Leverage FreeBSD's release engineering efforts

Use code from a -STABLE development branch of FreeBSD. These development branches are formally supported by FreeBSD's release engineering and security teams and comprise of tested code.

Donate code to reduce costs

A major proportion of the costs associated with developing products is that of doing maintenance. By donating non-critical code to the project, you benefit by having your code see much wider exposure than it would otherwise get. This in turn leads to more bugs and security vulnerabilities being flushed out and performance anomalies being identified and fixed.

Get support effectively

For products with tight deadlines, it is recommended that you hire or enter into a consulting agreement with a developer or firm with FreeBSD experience. The FreeBSD related employment mailing list is a useful communication channel to find talent. The FreeBSD project maintains a gallery of consultants and consulting firms undertaking FreeBSD work. The BSD Certification Group offers certification for all the major BSD derived OSes.

For less critical needs, you can ask for help on the project mailing lists. A useful guide to follow when asking for help is given in How to ask questions the smart way.

Publicize your involvement

You are not required to publicize your use of FreeBSD, but doing so helps both your effort as well as that of the project.

Letting the FreeBSD community know that your company uses FreeBSD helps improve your chances of attracting high quality talent. A large roster of support for FreeBSD also means more mind share for it among developers. This in turn yields a healthier foundation for your future.

Support FreeBSD developers

Sometimes the most direct way to get a desired feature into FreeBSD is to support a developer who is already looking at a related problem. Help can range from hardware donations to direct financial assistance. In some countries, donations to the FreeBSD project enjoy tax benefits. The project has a dedicated donations liaison to assist donors. The project also maintains a web page where developers list their needs.

As a policy the FreeBSD project acknowledges all contributions received on its web site.

4. Conclusion

The FreeBSD project's goals are to create and give away the source code for a high-quality operating system. By working with the FreeBSD project you can reduce development costs and improve your time to market in a number of product development scenarios.

We examined the characteristics of the FreeBSD project that make it an excellent choice for being part of an organization's product strategy. We then looked at the prevailing culture of the project and examined effective ways of interacting with its developers. The article concluded with a list of best-practices that could help organizations collaborating with the project.

Bibliography

[Carp1996] The Architectural Principles of the Internet B. Carpenter. The Internet Architecture Board.The Internet Architecture Board. Copyright® 1996.

[ComGuide] Committer's Guide The FreeBSD Project. Copyright® 2005.

[GoldGab2005] Innovation Happens Elsewhere: Open Source as Business Strategy Ron Goldman. Richard Gabriel. Copyright® 2005. Morgan-Kaufmann.

[Hub1994] Contributing to the FreeBSD Project Jordan Hubbard. Copyright® 1994-2005. The FreeBSD Project.

[McKu1999] Soft Updates: A Technique for Eliminating Most Synchronous Writes in the Fast Filesystem Kirk McKusick. Gregory Ganger. Copyright® 1999.

[McKu1999-1] Twenty Years of Berkeley Unix: From AT&T-Owned to Freely Redistributable Marshall Kirk McKusick. Open Sources: Voices from the Open Source Revolution O'Reilly Inc.. Copyright® 1993.

[Mon2005] Why you should use a BSD style license for your Open Source Project Bruce Montague. The FreeBSD Project. Copyright® 2005.

[Nik2005] A project model for the FreeBSD Project Niklas Saers. Copyright® 2005. The FreeBSD Project.

[Nor1993] Tutorial on Good Lisp Programming Style Peter Norvig. Kent Pitman. Copyright® 1993.

[Nor2001] Teach Yourself Programming in Ten Years Peter Norvig. Copyright® 2001.

[Ray2004] How to ask questions the smart way Eric Steven Raymond. Copyright® 2004.

[RelEngDoc] FreeBSD Release Engineering Murray Stokely. Copyright® 2001. The FreeBSD Project.

[1] FreeBSD's source repository contains a history of the project since its inception, and there are CDROMs available that contain earlier code from the CSRG.