How the FreeBSD Project Works

Robert N. M. Watson
rwatson@FreeBSD.org

FreeBSD Project

Computer Laboratory
University of Cambridge

1 Introduction

FreeBSD is a widely deployed open source operating system. [3] Found throughout the industry, FreeBSD is the operating system of choice for many appliance products, embedded devices, as a foundation OS for several mainstream commercial operating systems, and as a basis for academic research. This is distinct, however, from the FreeBSD Project, which is a community of open source developers and users. This paper discusses the structure of the FreeBSD Project as an organization that produces, maintains, supports, and uses the FreeBSD Operating System. As this community is extremely large, I approach this from the perspective of a FreeBSD developer. This necessarily captures the project from my perspective, but having had the opportunity to discuss the FreeBSD Project extensively with many people inside and outside the community, I hope it is also more generally applicable.

2 Introduction to FreeBSD

FreeBSD is an open source BSD UNIX operating system, consisting of a kernel, user space environment, extensive documentation, and a large number of bundled third party applications. It is widely used as an ISP server platform, including at well-known providers such as Yahoo!, Verio, New York Internet, ISC, Demon, and Pair. It is also widely used in part or in whole for appliances and embedded devices, including Juniper's JunOS, Nokia's IPSO, and for commercial operating system products, such as VXWorks and Mac OS X. The product of one of the most successful open source projects in the world, FreeBSD development work has focused on the areas of storage, networking, security, scalability, hardware support, and application portability.

The highly active FreeBSD development community centers on services offered via FreeBSD.org, which include four CVS repositories and a Perforce repository. These represent the life-blood of the development and documentation work of the Project. There are over 300 active developers working in CVS, which hosts the official development trees for the base source code, Ports Collection, projects tree, and documentation project. Significant project work also takes place in Perforce, which supports a heavily branched concurrent development model as well as guest accounts and external projects.

Another defining feature of the FreeBSD Project is its use of the liberal Berkeley open source license. Among features of the license are is remarkable simplicity (the license can be fully displayed in an 80x24 terminal window) and its ability to support derived works that are closed source, key to commercial and research adoption of FreeBSD.

3 What do you get with FreeBSD?

FreeBSD is a complete, integrated UNIX system. The core of FreeBSD is a portable multi-processing, multi-threaded kernel able to run on a variety of hardware platforms including Intel/AMD 32-bit and 64-bit processors, Intel's Itanium platform, and Sun's UltraSparc platform. FreeBSD is also able to run on several embedded platforms based on i386, ARM, and PowerPC; a MIPS port is also underway.

FreeBSD implements a variety of application programming interfaces (APIs) including the POSIX and Berkeley Sockets APIs, as well as providing a full UNIX command line and scripting environment. The FreeBSD network stack supports IPv4, IPv6, IPX/SPX, EtherTalk, IPSEC, ATM, Bluetooth, 802.11, with forthcoming support for SCTP. Security features include access control lists (ACLs), mandatory access control (MAC), security event auditing, pluggable authentication modules (PAM), and a variety of cryptographic services. FreeBSD ships with both workstation/server and embedded development targets, and comes with extensive user and programmer documentation.

FreeBSD also ships with ports of over 16,000 third party open- and closed-source software packages, providing programming and user interfaces such as X11, KDE, Gnome, OpenOffice, and server software such as Java, MySQL, PostgreSQL, and Apache.

4 The FreeBSD Project

The FreeBSD Project's success can be measured by the extremely wide deployment of FreeBSD-based systems. From root name servers to major web hosts, search engines, and routing infrastructure, FreeBSD may be found at most major service providers. FreeBSD is also the foundation for a number of commercial operating systems. The FreeBSD Project is more than just software, or even software development: it includes a global community of developers, port maintainers, advocates, and an extensive user community. Central to this community are the FreeBSD.org web site, FTP site, CVS repository, and mailing lists.

Several papers and studies have been written on the topic of the FreeBSD Project and its development process, including a papers by Richards [7], Jorgensen [4], and Dinh-Trong [1].

5 The FreeBSD Foundation

The FreeBSD Foundation is a non-profit organization based in Boulder, CO. By design, the Foundation is separate from the FreeBSD Project. When the Foundation was created, it was not clear that a non-profit supporting open source development was a viable concept. As such, it was important to the founders that the Foundation be a separate legal entity that would support the Project, but that the Project not be dependent on the long-term viability of a Foundation. It was also important to the founders of the Foundation that there be a differentiation between the people managing the monetary, legal, and administrative matters and those administering the software development work in the project. In practice, the Foundation has proved financially and administratively successful, and plays an important role in supporting the daily operation and long term success of the Project.

The FreeBSD Foundation is responsible for a broad range of activities including contract development (especially relating to Java), managing of intellectual property, acting as a legal entity for contractual agreements (including non-disclosure agreements, software licensing, etc), providing legal support for licensing and intellectual property issues, fund-raising, event sponsorship (including BSDCan, EuroBSDCon, AsiaBSDCon, and several FreeBSD developer summits a year), providing travel support for FreeBSD developers and advocates, negotiating collaborative R&D agreements, and more.

The FreeBSD Foundation is currently managed by a board of directors, and has one part-time employee who is responsible for day-to-day operation of the Foundation as well as sitting on the board. The board also consists of four volunteer members drawn from the FreeBSD developer community. The FreeBSD Foundation Board is in regular communication with other administrative bodies in the FreeBSD Project, including the FreeBSD Core Team.

The FreeBSD Foundation is entirely supported by donations, and needs your help to continue its work!

6 What We Produce and Consume

The FreeBSD Project produces a great deal of code: the FreeBSD kernel, user space, and the Ports Collection. But the FreeBSD Project does not produce "just source code". FreeBSD is a complete software product, consisting of software, distribution, documentation, and support:

- FreeBSD kernel, user space
- Ports collection, binary package builds
- FreeBSD releases
- FreeBSD manual pages, handbook, web pages, marketing material
- Architecture and engineering designs, papers, reports, etc
- Technical support, including answering questions and debugging problems
- Involvement in and organization of a variety of FreeBSD user events

This would not be possible without support of a larger community of users and consumers, who provide certain necessary commodities:

- Beer, wine, soda, chocolate, tea, and other food/beverage-related vices in significant quantity.
- Donated and sponsored hardware, especially in racks at co-location centers, with hands to help manage it.
- Bandwidth in vast and untold quantities.
- Travel grants, developer salaries, contracts, development grants, conference sponsorship, organization membership fees, etc.
- Thanks, user testimonials and appreciation, good press.
- Yet more bandwidth.

None of these has a trivial cost–by far the most important resource for the project is developer time, both volunteered and sponsored.

7 Who are the Developers?

FreeBSD developers are a diverse team, made up of members from 34 countries on six continents. They vary in age between 17 and 58, with a mean age of 32 and median age of 30; the standard deviation is 7.2 years. FreeBSD developers include professional systems programmers, university professors, contractors and consultants, students, hobbyists, and more. Some work on FreeBSD in a few spare hours in the evening once a week–others work on FreeBSD full time, both in and out of the office. FreeBSD developers are united by common goals of thoroughness and quality of work. Unlike many open source projects, FreeBSD can legitimately claim to have developers who have worked on the source base for over thirty years, a remarkable longevity that would be the envy of many software companies. This diversity of experience contributes to the success of FreeBSD, combining the pragmatic "real world problem" focus of consumers building products with the expertise of researchers working on the cutting edges of computer science research.

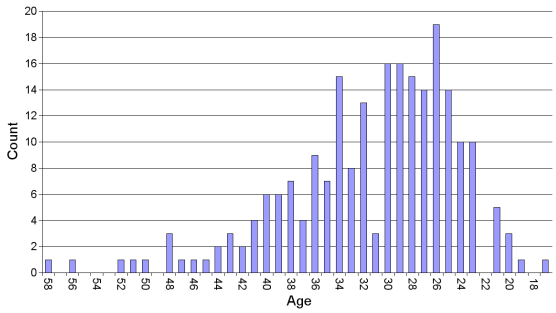

Figure 1: Age Distribution of FreeBSD Developers (2005)

8 FreeBSD Processes

The FreeBSD Project is successful in significant part because it encapsulates not just many experienced and highly competent individuals, but also because it has a set of well-defined development processes and practices that are universally accepted and self-sustaining.

- *Committer life cycle and commit bits* - The process by which new developers are inducted into the community and mentored as new members of the community is well-defined and successful.

- *Core Team* - Project leadership is selected and renewed via regular elections from the developer team as a whole, insuring both continuity, continued engagement, and fresh voices lead the project over time.

- *Mailing lists* - Through extensive and courteous use of mailing lists for almost all project communications over many years, consensus is almost universal in project decision making, and there is relatively little "stepping on toes" for a project that spans dozens of countries and time zones.

- *Web pages and documentation* - A well-designed and extremely complete set of web pages and documentation provide access to both the current condition and history of the project, from tutorial content for new users to detailed architectural information on the design of the kernel.

- *Groups/projects* - A hallmark of FreeBSD's success is the scalable community model, which combines the best of centralized software development with project-oriented development, allowing long-term spin-off projects to flourish while maintaining close ties and involvement in the central project.

- *Events* - The FreeBSD Project exists primarily through electronic communication and collaboration, but also through in-person developer and user events occurring continuously throughout the year. These include developer summits and involvement in both BSD-specific and general purpose conferences.

- *Honed development and release cycle* - With over ten years of online development and release engineering experience, the FreeBSD Project has pioneered many online development practices, combining professional software engineering approaches with pragmatic approaches to volunteer-driven open source development. One of the key elements of this approach is effective and highly integrated use of software development tools and revision control, including the use of multiple revision control systems, CVS and Perforce.

- *Centralized computing resources* - Also key to the success of the project has been the use of several globally distributed but centrally managed computing clusters, organized and maintained by project donors and a highly experienced system administration team. The FreeBSD.org infrastructure "just works", providing flawless support for the daily activities of the project.

- *Conflict resolution* - In any development project, but especially in widely distributed organizations, effective management of technical disagreements and conflicts is critical; the FreeBSD Project's history is full of examples of successful conflict resolution leading to both good technical and social outcomes.

8.1 FreeBSD Committers

A FreeBSD committer is, in the most literal sense, someone who has access to commit directly to the FreeBSD CVS repository. Committers are selected based on four characteristics: their technical expertise, their history of contribution to the FreeBSD Project, their clear ability to work well in the FreeBSD community, and their having made the previous three extremely obvious. Key to the induction of new committers is the notion of a mentor: this is an existing committer who has worked with the candidate over an extended period and is willing to both sponsor their candidacy and also act in a formal role in introducing them to the project. The mentor proposes the candidate to one of the Core Team, Port Manager, or Doceng, who respectively approve commit rights for the src tree, the ports tree, or the documentation tree. A typical proposal includes a personal introduction of the candidate, a history of their background and contribution, and volunteers to mentor them.

Once approved, typically by a vote, the new committer is given access to the FreeBSD.org cluster and authorized access to CVS. Mentorship does not end with the proposal: the mentor and new committer will have a formal ongoing relationship for several months, in which the mentor works with the new committer to review and approve all commits they will make, helps them circumnavigate the technical and social structure of the project. This relationship often continues informally in the long term, beyond the point where the mentor has "released" the new committer from mentorship. Typically, there is significant technical interest overlap between the proposing mentor and the new committer, as this will be the foundation on which familiarity with their work, as well as competence to review their work, will have been formed.

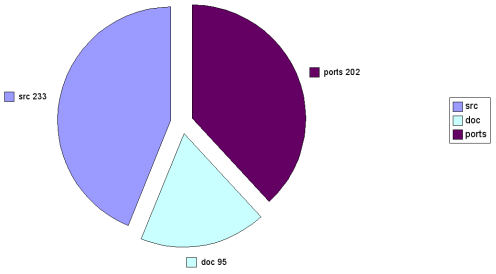

Figure 2: Number of FreeBSD committers by commit bit type (2005)

Committers often begin working in one of the various trees, and gradually spread to working in others. For example, it is not uncommon for documentation committers to expand the scope of their work to include source development, or for src developers to also maintain a set of application ports. Some of FreeBSD's most prolific and influential kernel developers have begun life writing man pages; "upgrading" a commit bit to allow access to new portions of the tree is a formal but lightweight process, in which a further proposal by a potential mentor is sent to the appropriate team for approval. As with an entirely new committer, a formal mentorship will take place, in which the new mentor takes responsibility for reviewing their commits during their earlier work with their new commit bit.

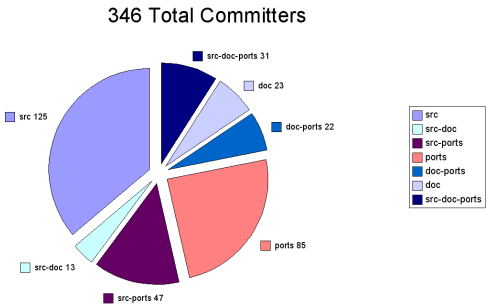

Figure 3: There is significant overlap, with many committers working in more than one area of the source tree. (2005)

8.2 FreeBSD Core Team

The FreeBSD Core Team is the nine-member elected management body of the FreeBSD Project, and is responsible for a variety of administrative activities. Historically, the Core Team consisted of a self-selected set of the leading developers working on FreeBSD; however, in 2000, the model was changed to an elected model in order to adopt a more sustainable model. Every two years, nominees from the FreeBSD committer team volunteer to be placed on the role, and a one month online election is held. The FreeBSD Core Team then appeals for and selects a volunteer to act as Core Secretary.

While the process of selecting the Core Team is well-defined, the precise responsibilities of the Core Team are not, and have evolved over time. Some activities are administrative in nature: organizing successive elections, assisting in writing and approving charters for specific teams, and approving new FreeBSD committers. Other activities are more strategic in nature: helping to coordinate developer activity, making sure that key areas are being worked in by cajoling or otherwise convincing developers they are important, and assigning authority to make significant (possibly contentious) architectural decisions. Finally,

the FreeBSD Core Team is responsible for maintaining and enforcing project rules, as well conflict resolution in the event that there is a serious disagreement among developers.

8.3 Ports Committers, Maintainers

The FreeBSD Ports Collection is one of the most active areas of FreeBSD work. At its heart, the ports tree is a framework for the systematic adaptation of third party applications to FreeBSD, as well as a vast collection of ported applications. In 2005, there were 158 ports committers working on 16,000 application ports. In addition to ports committers, the notion of a ports maintainer is also important: while ports committers are often involved in maintaining dozens or even hundreds of ports themselves, they also work to funnel third party porting work by over 1,500 ports maintainers into the ports tree. Particularly prolific maintainers often make good candidates for ports commit bits. With an average of 100 ports per committer and 11 ports per maintainer, the ports work is critical to the success of FreeBSD.

The Port Manager (portmgr) team is responsible for administration of the ports tree, including approving new ports committers as well as administering the ports infrastructure itself. This involves regression testing and maintaining the ports infrastructure, release engineering and building of binary packages across half a dozen hardware platforms for inclusion in FreeBSD releases, as well as significant development work on the ports infrastructure itself. Regression testing is a significant task, involving large clusters of build systems operating in parallel; even minor infrastructure changes require the rebuilding of tens of thousands of software packages.

8.4 Groups and Sub-Projects

The FreeBSD Project is a heavily structured and sizable organization with many special interest groups working in particular areas. These groups focus on specific technical areas, support, advocacy, deployment and support of FreeBSD in various languages and in different countries. Some sub-groups are formally defined by the project, and in some cases, have approved charters and membership. Others exist more informally, or entirely independent of the central FreeBSD.org infrastructure, shipping derived software products.

8.5 A FreeBSD Project Org Chart

While the concept of an organizational chart applies somewhat less well to a loose-knit volunteer organization than a traditional company, it can still be instructive.

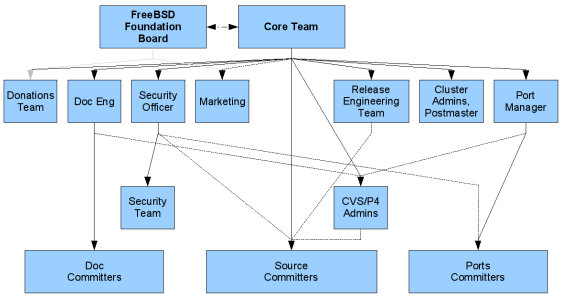

Figure 4: Lines in this FreeBSD Project Org chart represent more than just downward delegation of authority found in commercial organizations.

In a traditional organization chart, arrows would represent delegation of responsibility. In the FreeBSD Project organization chart, this is only partially true: typically arrows represent delegation of authority: i.e., the FreeBSD Core Team, the elected management body of the project has assigned authority, by means of voting to approve a written chart, for security advisory and other Security Officer activities to the Security Officer and Security Officer team. As the organization is volunteer-driven, delegation of of responsibility occurs up as much as down: the larger body of FreeBSD committers select a Core Team to take responsibility for a variety of administrative activities.

8.6 Derived Open Source Projects

FreeBSD provides components, and in some cases the foundation, of a large number of derived open source software projects.

- FreeSBIE, a FreeBSD-based live CD image
- m0n0wall, an embedded FreeBSD-based firewall package
- pfSense, an extensible firewall package based on m0n0wall
- PC-BSD, a workstation operating system based on FreeBSD
- Darwin, the open source foundation of the Mac OS X operating system, which includes both portions of the FreeBSD kernel and user space
- DesktopBSD, a workstation operating system based on FreeBSD
- DragonflyBSD, a FreeBSD-derived research operating system project

- FreeNAS, a FreeBSD-based network storage appliance project

In addition, FreeBSD code may be found in an even greater number of projects that software components developed in FreeBSD; this includes open source projects such as OpenBSD, NetBSD, and Linux systems.

8.7 Mailing Lists

Mailing lists are the life-blood of the project, and the forum in which almost all project business takes place. This provides a long term archive of project activities. There are over 40 public mailing lists hosted at FreeBSD.org, as well as a number of private mailing lists associated with various teams, such as the Core Team, Release Engineering team, and Port Manager team. Mailing lists serve both the developer and user communities. A great many other mailing lists relating to FreeBSD are hosted by other organizations and individuals, including regional user groups, and external or derived projects.

8.8 FreeBSD Web Pages

Web sites are a primary mechanism by which the FreeBSD Project communicates both internally and with the world at large. The main FreeBSD.org web site acts as a distribution point for both FreeBSD as software and documentation, but also as a central point for advocacy materials. Associated web sites for the mailing lists and mailing list archives, bug report system, CVSweb, Perforce, and many other supporting services are also hosted as part of the FreeBSD.org web site.

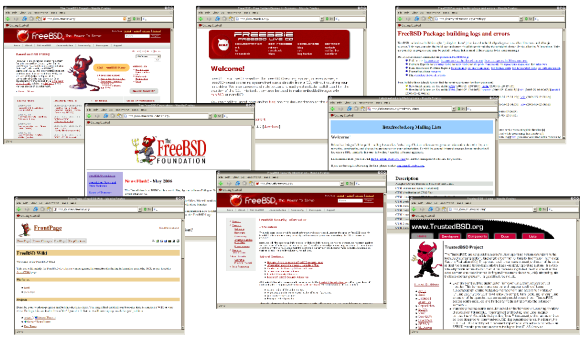

Figure 5: Web sites play an integral role in how the FreeBSD Project communicates with both users and contributors.

In addition, there are a number of project-specific web sites for FreeSBIE, TrustedBSD, PC-BSD, DesktopBSD, and others, which are linked from the main FreeBSD.org web site, but are separately authored and hosted.

8.9 Events

While electronic communications are used as the primary method of communication for most on-going work, there is no substitute for meeting people you are working with in-person. The FreeBSD Project has a presence at a great many technical workshops and conferences, such as USENIX and LinuxWorld, not to mention a highly successful series of BSD-related conferences, such as BSDCan, EuroBSDCon, AsiaBSDCon, NYCBSDCon, MeetBSD, and a constant stream of local user group and developer events.

As these conferences bring together a great many FreeBSD developers, there are often Developer Summits occurring concurrently, in which FreeBSD developers meet to present, discuss, hack, and socialize. Summits typically consist of a formal session containing both presentations and moderated discussion, and information activities, such as hacking and gathering at a bar or pub.

8.10 FreeBSD Development Cycle

FreeBSD is created using a heavily branched development model; in revision control parlance, this means that there is a high level of concurrent work occurring independently. The central FreeBSD src CVS repository contains a large number of branches; the main of these is the HEAD or CURRENT branch, where new features are aggressively developed.

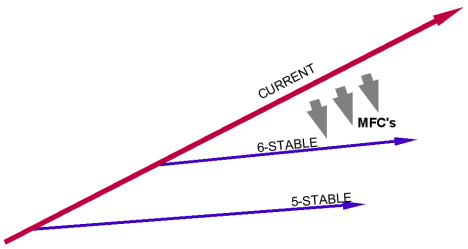

Figure 6: Branching is a key element of the FreeBSD development model: simultaneous work on several complete versions of FreeBSD at once allows changes to be merged from one branch to another as they gain stability, exposing them to successively wider testing and use.

A series of STABLE branches contains more conservative development, one per major release series, with changes being trickled from the CURRENT branch to

other branches as they stabilize; this process is referred to as "Merged From Current", or MFC. Minor releases are cut from STABLE branches at regular intervals, typically three to six months. Major releases are cut around every 18 months, although sometimes less frequently, and involve the creation of a new STABLE branch; this allows extremely large features, inappropriate for merge to a STABLE release series, to be released as part of new major (.0) releases.

In addition to the CURRENT and STABLE branches, RELEASE branches are used for release cycles as well as for security and errata patches following release.

7-current	cutting edge development
6-stable	active development with releases
5-stable	legacy branch with releases
4-stable	legacy branch

Branched development is also used extensively during early feature development. Due to limitations in CVS, discussed later, this work typically occurs in branches in the FreeBSD Perforce server.

8.11 FreeBSD Releases

Release engineering is one of the most tricky aspects of running any large software project, let alone a large-scale, volunteer-run open source project. The release team (RE) is responsible for the coordinating the combination of technical and technical engineering necessary to bring a FreeBSD release to fruition. With membership approved by the Core Team, RE is given significant leeway to steer the FreeBSD development process, including placing administrative limits on development in the tree (code slushes, freezes), performing CVS branching and tagging operations, not to mentioning begging and cajoling developers into doing that which is necessary to make a release possible.

As FreeBSD development is centered on revision control, the revision control operations involved in a release are important to understanding how releases occur. Releases occur in release branches, which are typically branched from a -STABLE development branch. In preparation for a release, development on the -STABLE branch is slowed to a more conservative set of changes in order that existing new work can stabilize. First a "code slush" occurs, in which new features are eschewed, but bug fixing and refinement occurs largely unhindered; any significant changes for the release require approval by the Release Engineering team during this period. After a period of slush, a "code freeze" is started, after which point commits to the tree may only occur with the specific approval of the release am. This change in process increases the level of review taking place for changes, as well as allowing the Release Engineering team to manage risk for the release as a whole.

A series of beta test releases will be made during the code freeze, in which major and minor problems are incrementally identified and corrected. Once the Release Engineering team is satisfied with the quality of the tree, branching of the release branch may occur, which can allow more active development on the -STABLE branch to resume. A series of release candidates is used to continue to refine the release, with successively more broad testing, especially of the install procedure, which sees less exposure during normal development. Once a final release candidate is created, the release itself may occur, and the release is tagged.

Coordinated with this process for the base tree is both a release process for the ports and documentation trees. Final third party package builds occur prior to the release candidate series, ensuring testing and compatibility after significant changes have been completed in the base source tree. The Port Manager team also places a slush and freeze on the ports tree, allowing testing of the packages together rather than in isolation. The documentation tree is likewise tagged as part of the release process; an important aspect of the release is preparation of the release documentation, including the release notes identifying changes in FreeBSD, finalization of translated versions, and updates to the web site and documentation to reflect the release.

The release branches continue to serve an important role after the tagging and release of a FreeBSD version. Once the Release Engineering team believes that there is no risk of a re-roll of the release due to a last minute issue, it will transfer ownership of the branch to the Security Officer team, which will then maintain security patches against the release in that branch. The Release Engineering team may also coordinate the addition of errata patches to the branch for major stability or functional problems identified after the release. Freezes requiring approval of the Release Engineering or Security Officer teams are not released on release branches.

The FreeBSD 6.1 release process is fairly representative, in that it contained the typical snags and delays, but produced a very technically successful and widely deployed release:

25 Jan 2006	Schedule finalized
31 Jan 2006	Code freeze begins
5 Feb 2006	Ports schedule, announced
5 Feb 2006	6.1-BETA1
19 Feb 2006	6.1-BETA2
23 Feb 2006	Ports tree frozen
3 Mar 2006	6.1-BETA3
6 Mar 2006	Doc tree slush
14 Mar 2006	6.1-BETA4; ports tagged
5 Apr 2006	RELENG_6_1 branch
10 Apr 2006	6.1-RC1
17 Apr 2006	Doc tree tagged, unfrozen
2 May 2006	6.1-RC2
7 May 2006	Release tagged
7 May 2006	Build release
8 May 2006	6.1-RELEASE released

Major (.0) releases occur in a similar manner to minor releases, with the added complexity of creating a new -STABLE branch as well as a new release branch. As this occurs quite infrequently, often as much as several years apart, the process is more variable and subject to the specific circumstances of the release. Typically, the new -STABLE branch is created after a long period of code slush and stabilization in the -CURRENT branch, and occurs well in advance of the formal release process for the .0 release. Critical issues in this process include the finalization of application binary interfaces (ABIs) and APIs for the new branch, as many ABIs may not be changed in a particular release line. This includes library version updates, kernel ABI stabilization for device drivers, and more.

Incremental releases of FreeBSD, such as the 6.1 and 6.2 releases, largely require appropriately conservative strategies for merging changes from the CURRENT branch, along with some amount of persuasion of developers to address critical but less technically interesting issues. Typical examples of such issues are device driver compatibility issues, which tend to rear their heads during the release process as a result of more broad testing, and a few individuals bravely step in to fix these problems.

Larger releases, such as 3.0, 4.0, 5.0, and 6.0, require much more care, as they typically culminate several years of feature development. These have been handled with varying degrees of success, with the most frequent source of problems the tendency to overreach. While the FreeBSD 4.0 and 6.0 releases were largely refinements and optimizations of existing architecture, the FreeBSD 3.0 and 5.0 releases both incorporated significant and destabilizing architectural changes. Both resulted in a series of incremental releases on a STABLE branch that did not meet the expectations of FreeBSD developers; while these problems were later ironed out, they often resulted from a "piling on" of new features during an aggressive CURRENT development phase.

The success of the FreeBSD 6.x release series has been in large part a result of a more moderated development and merge approach, facilitated by the heavy use of Perforce, which allows experimental features to be maintained and collaborated on without merging them to the CVS HEAD before they are ready. Prior to the use of Perforce, experimental features were necessarily merged earlier, as there were not tools to maintain them independently, which would result in extended periods of instability as the base tree ceased to be a stable platform for development. The more mature development model leaves the CVS HEAD in a much more stable state by allowing a better managed introduction of new features, and actually accelerates the pace of development by allowing avoiding slowdowns in concurrent development due to an unstable base.

8.12 Revision Control

Most major technical activities in the project are centered on revision control. This includes the development of the FreeBSD source code itself, maintenance of the tends of thousands of ports makefiles and metadata files, the FreeBSD web site and documentation trees (including the FreeBSD Handbook), as well as dozens of large-scale on-going projects. Historically, FreeBSD has depended heavily on CVS, but has both extended it (via cvsup), and made extensive use of Perforce as the project has grown. The FreeBSD Project is now actively exploring future revision control options.

8.12.1 Revision Control: CVS

CVS, or the Concurrent Versions System, is the primary revision control system used by the FreeBSD Project, and holds the authoritative FreeBSD source trees, releases, etc. [2] This repository has over twelve years of repository history. The FreeBSD CVS repository server, repoman.FreeBSD.org, actually holds four separate CVS repositories:

/home/ncvs	FreeBSD src
/home/pcvs	FreeBSD ports
/home/dcvs	FreeBSD documentation
/home/projcvs	FreeBSD project

The FreeBSD Project supplements CVS in a variety of ways; the most important is cvsup, which allows high-speed mirroring and synchronization of both the CVS repository itself, as well as allowing CVS checkouts without use of the heavier weight CVS remote access protocol. This permits the widespread distribution of FreeBSD, as well as avoiding concurrent access to the base repository, which with CVS can result in a high server load. Most developers work against local CVS repository mirrors, only using the central repository for check-in operations.

Over time, the technical limitations of CVS have become more apparent; cvsup significantly enhances the scalability of CVS, but other limits, such as the lack of efficient branching, tagging, and merging operations have become more of an issue over time.

8.12.2 Revision Control: Perforce

While CVS has served the project extremely well, its age is showing. CVS fails to offer many key features of a distributed version control system, nor the necessary scalability with respect to highly parallel development. To address these problems, the FreeBSD Project has deployed a Perforce server, which hosts a broad range of on-going "projects" derived from the base source tree. [6] The most important feature that Perforce brings to the FreeBSD Project is support for highly branched development: it makes creating and maintaining large-scale works in progress possible

through lightweight branching and excellent history-based merging of changes from parent branches to children.

Currently, most major new kernel development work is taking place in Perforce, allowing these projects to be merged to the base tree as they become more mature, avoiding high levels of instability in the CURRENT branch. Perforce also makes collaboration between developers much easier, allowing developers to monitor each other's works in progress, check them out, test them, and modify them. Projects that have been or are being developed in Perforce include SMPng, KSE, TrustedBSD Audit, TrustedBSD MAC, SEBSD, superpages, uart, ARM, summer of code, dtrace, Xen, sun4v, GEOM modules, CAM locking, netperf, USB, ZFS, gjournal, and many others. CVS remains the primary and authoritative revision control system of the FreeBSD Project, with Perforce being reserved for works in progress, but it plays a vital role in the growth of the project, so cannot be ignored in any serious consideration of how the project operates.

8.12.3 Revision Control: The Future

The FreeBSD Project is in the throes of evaluating potential future distributed version control systems as a potential successor to CVS and Perforce, with the goal of subsuming all activity from both into a single repository. The Project's requirements are complicated, both in terms of basic technical requirements, as well as being able to support our development processes and practices. Primary of these requirements is that the entire current CVS repository and history be imported into the new repository system, a task of non-trivial complexity, and that it support the new branched development model used heavily in Perforce. Another important consideration is continued support for the cvsup infrastructure for the foreseeable future.

8.13 Clusters

The FreeBSD Project makes use of several clusters scattered around the world, typically located at co-location centers. These clusters are possible due to the generous donations of companies using FreeBSD. One of the most important aspects of these donations is that they are not just significant donations of servers or rack space, but donations of administrative staff time and expertise, including hands to rearrange and handle new and failing hardware, reinstall and update systems, and help troubleshoot network and system problems at bizarre hours of the day and night.

8.13.1 FreeBSD.org cluster

While there are several FreeBSD Project clusters, The FreeBSD.org Cluster is hosted in Santa Clara by Yahoo!, and is home of many of the most critical systems making up the FreeBSD.org domain.

Mail servers	hub, mx1, mx2
Distribution	ftp-master, www
Shell access	freefall, builder
Revision control	repoman, spit, ncvsup
Ports cluster	pointyhat, gohans, blades
Reference systems	sledge, pluto, panther, beast
Name server	ns0
NetApp filer	dumpster

All of these systems have been made available through the generous donations of companies supporting FreeBSD, such as Yahoo!, NetApp, and HP. The systems are supported by remote power, serial consoles, and network switches.

8.13.2 Other Clusters

The FreeBSD.org cluster hosted at Yahoo! is not the only concentration of FreeBSD Project servers. Three other major clusters of systems are used by the FreeBSD Project:

- The *Korean ports cluster* hosted by Yahoo! in Korea provides a test-bed for ports work.
- *allbsd.org* in Japan provides access to many-processor Sun hardware for stress and performance testing.
- The *Sentex cluster* hosts both the FreeBSD Security Officer build systems, as well as the Netperf cluster, a network performance testing cluster consisting of a dozen network booted systems with gigabit networking. This cluster has also been used to test dtrace, hwpmc, and ZFS.
- The *ISC cluster* hosts half of FreeBSD.org, as well as a large number of ports building systems, the FreeBSD.org Coverity server, test systems, and more.

8.14 Conflict Resolution

Conflict resolution is a challenging issue for all organizations, but it is especially tricky for volunteer organizations. FreeBSD developers are generally characterized by independence, a good sense of cooperation, and common sense. This is no accident, as the community is self-selecting, and primary criteria in evaluating candidates to join the developer team are not just technical skills and technical contribution, but also the candidate's ability to work successful as part of a larger global development team. Conflict is successfully avoided by a number of means, not least avoiding unnecessary overlap in work areas and extensive communication during projects that touch common code.

Despite this, conflicts can and do arise: some consist purely of technical disagreements, but others result from a combination of the independence of spirit of FreeBSD developers and the difficulty of using solely

online communications to build consensus. Most conflicts are informal and self-resolving; on the rare occasion where this is not the case, the FreeBSD Core Team is generally responsible for mediating the conflict. For purely technical disagreements, reaching a decision by careful consideration (and fiat) is often successful, relying on the elected authority of the Core Team to make a final decision. As technical disagreements are often only the trigger in more serious conflicts, the Core Team typically selects a mediator (usually a Core Team member) to help work to improve communications between the disagreeing parties, not just pick a "right" technical solution.

8.15 Bike sheds

"Bike sheds" are a very special kind of conflict found, most frequently, in technical communities. First described by Parkinson in a book on management, the heart of the issue of the bike shed lies in the observation that, for any major engineering task, such as the designing of a nuclear power plant, the level of expertise and investment necessary to become involved is so significant that most contributions are productive; however, the building of a bike shed is something that anyone (and everyone) can, and will, express an opinion on. [5] Strong opinions prove easiest to have on the most trivial details of the most unimportant topics; recognizing this problem is key to addressing it. Bike sheds, while not unique to FreeBSD, are an art-form honed to perfection by the project. Since they have become better understood, they have become much easier to ignore (or dismiss once they happen). This terminology has now been widely adopted by many other open source projects, including Perl and Subversion.

9 Conclusion

The FreeBSD Project is one of the largest, oldest, and most successful open source projects. Key to the idea of FreeBSD is not just software, but a vibrant and active online community of developers, advocates, and users who cooperate to build and support the system. Several hundred committers and thousands of contributors create and maintain literally millions of lines of code in use on tens of millions of computer systems. None of this would be possible without the highly successful community model that allows the FreeBSD Project to grow over time, as well as permitting other projects to build on FreeBSD as a foundation.

References

[1] DINH-TRONG, T. T., AND BIEMAN, J. M. The FreeBSD Project: A Replication Case Study of Open Source Development. *IEEE Transactions on Software Engineering 31*, 6 (2005).

[2] FREE SOFTWARE FOUNDATION. cvs - Concurrent Versions System. http://www.nongnu.org/cvs/.

[3] FREEBSD PROJECT. FreeBSD Project home page. http://www.FreeBSD.org/.

[4] JORGENSEN, N. Putting it all in the trunk: incremental software development in the FreeBSD open source project. *Information Systems Journal 11*, 4 (2001), 321–336.

[5] PARKINSON, C. N. *Parkinson's Law; or, the Pursuit of progress*. John Murray.

[6] PERFORCE SOFTWARE. Perforce, the Fast Software Configuration Management System. http://www.perforce.com/.

[7] RICHARDS, P. eXtreme Programming: FreeBSD a case study. In *UKUUG Spring Conference and Tutorials: Conference Proceedings* (2006), UKUUG.