FreeBSD™
FOUNDATION

FreeBSD Foundation Proposal Submission Guidelines

Version 1.2
June 1, 2017

Contents

Chapter 1

Proposal Submission

Proposals may be submitted to the FreeBSD Foundation for work relating to any subsystems or infrastructure within the FreeBSD Operating System or Project. Proposals will be evaluated based on desirability, technical merit, and cost-effectiveness.

A proposal may be shared with the FreeBSD Foundation for assistance and refinement prior to official submission. In this case the proposal must be clearly identified as a preliminary submission, and specific reference should be made to incomplete or provisional information.

Proposals must address all of the items below. A sample proposal may be found in Appendix A.

1.1 Project Title

Choose a brief but memorable title for the proposal. Choosing a good title at the time of submission promotes consistency when the FreeBSD Foundation describes the project in newsletters, status reports, and other communications.

1.2 Contact Information

The proposal must include:

- Name
- Company name, if applicable
- Email address
- Mailing address
- Phone number

A short biography of the submitter should also be included, describing the submitter's previous experience with both FreeBSD and the subject of the proposal, as well as the submitter's prior involvement in the FreeBSD Project and community.

It is generally expected that individual submitting the proposal will perform the work described in the proposal. An explanation must be provided if this is not the case.

1.3 Project Summary

The proposal should include a summary of about 50 to 100 words that briefly describes the high-level goals and benefits of the project. The FreeBSD Foundation may use this in its newsletter or other publications.

1.4 Project Description

The objective of the project proposal is to identify what is to be done, explain why it needs to be done, and convince the FreeBSD Foundation's project committee of the following:

- The project will provide a valuable improvement to the FreeBSD project.
- The project will benefit a reasonable portion of the FreeBSD community.
- The submitter is qualified to do this project.
- The submitter has identified someone who can provide technical oversight.
- The submitter has the resources needed to complete the project within the stated time and constraints.

1.4.1 Deliverables

Describe the deliverables to be produced as part of the proposed project. This will commonly include source code and related documentation such as manual pages, but may also include benchmark data, technical reports, or other artifacts.

1.4.2 Development Process

Describe the planned development process to be used for the project, including revision control system and repository, planned project branch usage, and any special software tool chain components.

1.4.3 Testing

Sponsored software projects are expected to include a testing component, which will likely include some combination of unit tests, integration tests, and system tests. Unit tests should use the kyua framework and integrate into FreeBSD's standard test infrastructure installed in /usr/tests.

1.4.4 Documentation

Documentation is required for all new functionality. This will typically require at a minimum a man page. New features likely require a chapter in the FreeBSD handbook. Interactive applications and developer tools may have a built-in help system. One or more technical reports may be appropriate for certain projects.

1.5 Technical Monitor

Project proposals shall identify a technical monitor, who will provide oversight and review. The monitor must be familiar with the application domain of the proposal. Ideally, the monitor should also be familiar with the policies and process of the FreeBSD Project, although this will not always be possible.

Technical monitors are very important to the project development process, and lend accountability. Large or complex projects may include multiple technical monitors, each responsible for a different aspect of phase of the proposed project.

The technical monitor will typically be a volunteer from the FreeBSD community, or a FreeBSD Foundation staff member. An external technical monitor may be identified for projects where specific domain expertise is required and is not available within the community or Foundation. If this technical monitor requires compensation, that cost must be included within the overall project costs.

1.6 Project Costing

Include the full cost of the project, including your fees, reviewer compensation, and any applicable taxes and ancillary fees. If the project has optional components, please indicate them with separate subtotals. The base project must remain a viable and useful addition to FreeBSD if optional components are excluded.

The Foundation does not publish guidelines on overall project costs (or hourly rates) and evaluates each proposal on its own merit. Grants are awarded in support of work that the proposer wishes to take on to improve the FreeBSD project. As a result, the proposed amount is typically not comparable with rates in a commercial consulting arrangement.

1.7 Timeline and Milestones

Include a table with the proposed project start date and the expected completion date for each milestone or deliverable. Include a per-milestone costing if the project includes interim payments.

Please specify any flexibility that may exist in the schedule.

1.8 Acceptance Criteria

Acceptance Criteria are a set of pre-established requirements that must be successfully completed for a milestone and/or for the overall project to be considered complete. Acceptance criteria will generally include functional attributes, and may include performance, interoperability, reliability, and others attributes.

Appendix A

Sample Proposal

```
Integrate the LLD linker into FreeBSD

Contact Details
Name: John Smith
Company: JS Consulting LLC
Email: johnsmith@example.com
Mailing Address:
  123 Fake Street
  Springfileld OR 97477
  USA
Phone: +1 212 555 3253

Bio
===

John started using FreeBSD in 2003 as a member of the operating system
software team at Appcom Industries. He submitted his first patch to the
project in 2006 (committed as r190605) and became a committer in 2009.
Recently he has been heavily involved with a variety of tool chain components
in FreeBSD, and has commit privileges to the LLVM project as well.

Executive Summary
=================

FreeBSD’s standard tool chain is consists of a mix of leading-edge components
like the Clang compiler, and painfully outdated ones that hold back progress
in other areas, such as a version of the GNU linker from 2007. This project
will address one of the most pressing of these issues, by updating the FreeBSD
system linker to LLD, the new ELF linker in the LLVM family.

Project Description
===================

Since its inception FreeBSD has relied on the GNU tool chain both for building
the base system, and for a number of tools provided in the base system.
Updates to the base system tool chain eventually stalled for a number of
reasons including the GNU projects’ migration to the GPLv3 in 2007.

FreeBSD began the process of migrating to a modern tool chain by importing
the Clang/LLVM compiler in 2010. In January 2014, FreeBSD 10.0 was released
with Clang as the system compiler, no longer including an outdated GCC in the
release.

Work on integrating LLDB, the debugger in the LLVM family of projects, is
currently ongoing and it is expected to become the system debugger for
```

FreeBSD 11.0.

A number of outdated tool chain components remain in the FreeBSD base system. These are primarily components of GNU binutils, and of these the GNU linker, ld, is the most pressing in need of replacement.

FreeBSD currently uses ld from GNU binutils 2.17.50, the last GPLv2 version. This linker performed adequately for the size and complexity of software in 2007, but requires excessive amounts of memory and time to link large and complex C++ software common today. It also lacks support for useful features and new CPU architectures.

These issues cause a lot of grief for developers within the FreeBSD project, and those involved in other projects who wish to have their software run on FreeBSD. In the worst case a severely outdated tool chain leads to developers abandoning FreeBSD as a target platform altogether.

LLD is the linker from the LLVM family of projects. While it is still undergoing very active development, it is on track to become a viable base system linker for FreeBSD.

In this project I will work with the upstream LLD developers to first ensure that LLD is well supported and fully functional on FreeBSD, and then import LLD into the FreeBSD base system. LLD will be integrated into the build infrastructure and I will work with the ports team to handle the migration from GNU ld as the default ports linker.

At a basic level some modernization of the FreeBSD base system tool chain is required in order for FreeBSD to remain a viable target in the open-source ecosystem of today. This specific project will not only meet that requirement, but will also make FreeBSD an attractive target for moving the state of the art in open source linkers forward.

Deliverables
============

1. A list of linker scripts, options, and formats used by the base system userland and kernel builds, in order to guide the upstream LLD development process.
2. A set of patches integrated in the upstream LLD repository to enable support for the items identified in deliverable 1, including regression tests.
3. A version of LLD imported in the base system, installed as a standalone non-default linker (e.g., ld.lld)
4. Build infrastructure changes to allow the build to use LLD instead of GNU ld (e.g., by setting WITH_LLD=YES in /etc/src.conf).
5. In collaboration with the ports team, a set of changes to the ports tree to use LLD for all ports where it is feasible, allowing the use of GNU ld from the binutils port for others.
6. A transition plan resulting in the migration to LLD and the deprecation and removal of the in-tree GNU ld.

Development Process
===================

The first stage of development will be done in collaboration with the upstream LLD community. My changes will be developed in a git branch cloned from the official LLD git mirror, and push changes on an ongoing basis to my public GitHub repository at https://www.github.com/example/lld. These patches will be submitted to LLVM's Phabricator code review tool as they become ready, and I will commit them to the official repository upon approval.

The integration project will be developed in a git branch, cloned from the FreeBSD Project's git mirror at https://www.github.com/freebsd/freebsd. Work in progress will be pushed on an ongoing basis throughout the project to facilitate collaboration and testing.

Once ready the work will be brought into FreeBSD head by first importing LLD into a Subversion vendor branch and merging to the contrib subdirectory, as described in the developer's handbook. The git branch will then be rebased on this baseline, and the changes brought across into Subversion.

Testing
=======

As with the rest of Clang and LLVM, upstream LLD has a comprehensive test suite. Tests will be added to this infrastructure where necessary so that all features and modes required by FreeBSD are tested. All FreeBSD-specific behavior will also have test coverage.

The upstream test suite will be integrated into FreeBSD's test infrastructure so that the standard "kyua test" invocation will execute the LLD tests.

A broader call for community testing will be made once LLD is imported into the base system with a src.conf knob to use it.

A ports exp-run will also be requested at this time. Ports that fail to build will have an annotation added to their Makefile to force the use of GNU ld. Feedback about the missing functionality triggering the build failure will be provided to the upstream LLD community.

Documentation
=============

Man pages

Upstream LLD does not currently have a manual page. One will be written as part of this project for FreeBSD, with an offer to commit it to the upstream repository.

Developer's Handbook

An update to the FreeBSD Developer's Handbook will be included in this project, and will describe both LLD and its integration into the broader FreeBSD tool chain. For example, reference will be made to to Clang's -fuse-ld option for selecting the linker to use.

Technical Monitor
=================

FreeBSD Foundation project manager Ed Maste will be the technical monitor for this work.

Timeline and Milestones
=======================

M1	January 15, 2018	Identification of all linker scripts, features and options used in the FreeBSD base system.
M2	March 30, 2018	Changes integrated into upstream lld
M3	April 15, 2018	Import of upstream lld snapshot into FreeBSD
M4	April 30, 2018	FreeBSD build infrastructure in place (bmake)
M5	May 30, 2018	Ports that fail to build with LLD set to use GNU ld.
M6	June 15, 2018	Migration to LLD as the default linker.
M7	TBD	In-tree GNU ld 2.17.50 disconnected from the build.

Project Costing
===============

For this project grant I propose a flat rate of $1000 USD per milestone.