GNU Guix: Syntax and semantics of systemd units in the Shepherd

Alberto Eleuterio Flores Guerrero

Table of Contents

1. Name

Alberto Eleuterio Flores Guerrero

2. Email address

I have two main emails which I am using for different purposes:

For Google: albertoefg@gmail.com

For GNU mailing lists: albertoefg@posteo.mx

3. Title

GNU Guix: Syntax and semantics of systemd units in GNU Shepherd

4. Summary

The main objective of this project is to extend the capabilities of GNU Shepherd to be able to handle systemd unit files, specifically `service, socket` and `device` unit files.

The first goal of this project consists in extending Shepherd to match the capabilities of `systemd.service` unit files. As part of this work, the Shepherd will be extended with kernel Control Groups (`cgroups`) support for systems that support it.

The second part will consist in implementing other types of systemd units, in particular `systemd.socket` and `systemd.device`.

Finally, GNU Shepherd will be extended to parse systemd.unit files, with a procedure to load a `.service` file and return a `<service>` object, once all the previous work is done.

5. Benefits

Right now systemd is used by some of the biggest GNU/Linux distributions, this includes the FSF approved distributions Parabola, Trisquel and PureOS. It is pretty clear that systemd has the biggest user share of the Init systems (although systemd is more than just an init system).

With this in mind, adding the ability to handle systemd unit files to GNU Shepherd will have this benefits:

1. GNU Shepherd will be extended to have a bigger API than it currently has, making it more powerful.
2. Users will be able to use their custom systemd unit files with little or no effort on GNU Guix. This will also ease the work of migrating from most distributions to GNU Guix.
3. More distributions could be interested in adopting GNU Shepherd as an alternative to systemd.

5.1. Considerations

Systemd is a project outside of GNU Guix and GNU Shepherd, it has been under heavy development in recent years and the decisions by the developers are sometimes criticized. Changes might be expected and compatibility might be broken from time to time.

I understand this, and that my project will require continuous development and maintenance after the end of Google Summer of Code, I am willing to work on this for as long as possible.

6. Objectives

The main focus of my project will be to work on the GNU Shepherd internals, so it has all the functionality required to handle systemd service, socket and device unit files, and later work on the parsing of this files.

1. Extend GNU Shepherd capabilities to handle systemd.service files.

   1.1 Add the appropriate mechanisms to GNU Shepherd API according to the systemd.unit and systemd.service files.

   1.2 Extend GNU Shepherd to be able to manage control groups (cgroups), according to the configurations of systemd unit files, which can specify limits for services and resources.

   1.3 Later, extend GNU Shepherd to handle other systemd.unit files, specifically systemd.socket and systemd.device.
2. Add a library to parse systemd.service files and return an instance of the <service> class.

6.1. Notes

All the implementations of systemd unit files and cgroups will be according to the Freedesktop specification.
https://www.freedesktop.org/wiki/Software/systemd/ControlGroupInterface/
https://www.freedesktop.org/software/systemd/man/systemd.unit.html
https://www.freedesktop.org/software/systemd/man/systemd.service.html

7. Deliverables

1. The modules `(shepherd)` and `(shepherd service)` should be extended to match the functionality of systemd.service, systemd.socket and systemd.device files.

   1.1. GNU Shepherd support to create and manage cgroups according to the configuration parameters on systemd unit files.
2. A module written to parse systemd.unit files specifically systemd.service, systemd.socket and systemd.device.
3. Appropriate unit tests in the tests directory.
4. The documentation for GNU Shepherd is written in the texinfo format in the file shepherd.texi, this file will be updated to reflect the changes and new capabilities of GNU Shepherd.

8. Plan

8.1. Before announcements (Today - May 4)

I would like to work during this period in my Scheme abilities, solve bugs for Shepherd and Guix, get familiar with the coding standards, environment setup and workflow.

8.2. Community Bonding Period (May 4 - June 1)

In case I get accepted to work on this project, I would use this time to:

1. Planning with my help from my mentor the design of the modules, functions and slots to be implemented during the project.
2. Compare systemd and GNU Shepherd API's.
3. Write function signatures for the functions that will be implemented on GNU Shepherd.
4. Become familiar with GNU Shepherd internals.
5. Work on my GOOPS abilities.

8.3. Coding (June 1 - August 24)

8.3.1. Period 1 (June 1 - June 29)

1. Implement on modules `(shepherd)` and `(shepherd services)` all the missing mechanisms of the `systemd.service` API, so in the future stages it can parse a `.service` file and return a `<service>` object.
2. Implement `cgroups` support according to the configurations of systemd unit files, which can specify limits for services and resources.
3. Write unit tests and documentation for the work done.

At this point Shepherd will have a bigger API and extended capabilities, matching the `systemd.service` API, however the proper parsing of this files would be left for a latter stage.

https://www.freedesktop.org/software/systemd/man/systemd.service.html

8.3.2. Period 2 (July 1 - July 27)

1. Implement the missing mechanisms to handle `systemd.socket` files, which encode information about an IPC or network socket or a file system FIFO controlled and supervised by systemd, for socket-based activation.

   https://www.freedesktop.org/software/systemd/man/systemd.socket.html
2. Implement the missing mechanisms to handle `systemd.device` files,which in turn handle a device unit as exposed in the sysfs/udev(7) device tree. Note: This type of files have no specific options.

   https://www.freedesktop.org/software/systemd/man/systemd.device.html

3. Write unit tests documentation.

At this point Shepherd should have most of the functionality required and have a bigger API. This period might be also used to wrap any details not finished in the previous one.

8.3.3. Period 3 (July 27- August 24)

1. Write a module to parse `systemd.service` files and return them as a `<service>` object.
2. Make sure tests are comprehensive and documentation is finished.
3. Double check that everything is working and compiling.

At this point Shepherd has all the functionality required to parse `systemd.service` files, and handle them as `<service>` object.

8.4. Code submissions

Everything should compiled, finished, and sent to GNU Shepherd git repository on savannah.gnu.org.

8.5. After GSOC

I would like to keep working on extending GNU Shepherd, improving the documentation, and making sure everything keeps working in case there are changes in systemd.

8.6. How will everybody know whether things are on-track at the half-way evaluation point?

For my code repository I can set up a Gitlab (or any git repository) or send the code I write everyday to the mailing list. Whichever way my mentor feels is more appropriate.

For the milestones, I've listed on 8.3.1, 8.3.2 and 8.3.3 the most important goals of each period. There might be some necessary adjustments to the timeline, but this should be fully discussed with my mentor.

I intend to be on the #guix freenode channel everyday, as I have been doing already for a while. I will also try to ask as many questions as I can to make sure my project is in the right track.

8.7. Other activities

Part of my school semester will overlap, although my school activities are mostly on weekends so it should not interfere with my work of GSOC.

9. Communication

I can communicate using the Mailing Lists of GNU Guix. I have sent a few patches already.

I can communicate in IRC with the nickname Blackbeard and with the Matrix account @blackbeard:matrix.eunichx.us

10. Qualification

10.1. Why did this project appeal to you?

- ➔ It is a GNU project
- ➔ I would love to work on a Scheme project.
- ➔ I am a user of both Guix System, and Guix package manager. Sometime in the past I installed GuixSD and thought "it would be easier for me to be able to use systemd files" because I didn't know how to use Shepherd.
- ➔ I think this project could benefit users not used to Scheme and who want an easy way to enjoy the benefits of Guix without taking much of their time.

10.2. How will you benefit from it?

I will learn about GNU Shepherd internals and will be able to contribute more to GNU Guix and become better at programming, specially in Scheme.

10.3. Why are you particularly suited to work on this?

I am not a Computer Science student and I might not be really qualified, however I particularly like Lisp, I am recognized as a small contributor in the Acknowledgments of the book "How to Design Programs 2 Edition" (as Alberto E. F. Guerrero) a book that teaches functional programming and test driven development in Racket. I am also a proficient Emacs user with a huge configuration and custom modes.

I've also read and worked through Structure and Interpretation of Computer Programs, The Land of Lisp, among others. This means I can be an independent learner and I would enjoy working in this project. I've also sent a couple of patches to GNU Guix already and I try to help users on the #Guix IRC channel as much as I can.

10.4. Background

In the personal side, I am a promoter of Software Freedom.

Currently, I am a student of a master of Law of IT, because I want to promote and advocate for the human rights in the digital world, to do that, I am on a path on learning as much as possible about software as well as about the law, so I can be an expert on both the technical and legal sides.

I've even met Richard Stallman in person a few years ago when I assisted to one of his conferences in my country.