Porting GuixSD to GNU/Hurd

Manolis Ragkousis
manolis837@gmail.com

March 17, 2016

1. Summary

In this project, we would like to port the Guix Software Distribution to GNU Hurd. By the end of the project, Guix will be able to handle and use the available Hurd mechanisms, have the required tools to produce a working, bootable, system vm/image with GNU Shepherd as the init daemon and offer the same functionality as you would expect from booting into a GNU/Linux GuixSD system.

2. The Project

The project consists of four main stages

1. Modify Guix so it will be able to create and mount the file-system needed to boot into a system with Hurd at its core.
2. Modify Guix so it can produce a working image, while isolating any cases of Linux assumptions.
3. Successfully boot into one such system using GNU Shepherd with pid 1.
4. Modify the new Guix system to take advantage of Hurd specific mechanisms.

Currently the tools Guix uses to interact with the filesystem exist inside the `(guix build syscalls)` module. This module provides bindings to libc's syscall wrappers, which are only available on a GNU/Linux system. In order to offer the same functionality on a GNU/Hurd system we must first write Guile bindings for the relevant Hurd functions, like ‘file_set_translator’ in hurd/fs.defs for example. This module will be called `(guix build hurd)`. This allows us to re-implement the functionality of the ‘settrans’ command, as described here, in Scheme and use that in place of ‘mount()’, where applicable.

Additionally, we need to make sure that all modules in `(guix build *)` and `(gnu build *)` can offer the same functionalities on a GNU/Hurd system. For example `(gnu build vm)` relies on QEMU's '-kernel' command-line option, which is Linux-specific. On the other hand, in the case of modules like `(gnu build linux-boot)` or `(gnu build linux-initrd)`, which by design are Linux-specific, we will need to provide a separate module with equivalent functionality. `(gnu system *)` modules will require changes as well, in order to be able to use modifications that will happen in the `(guix build *)` and `(gnu build *)` modules.

At this point we will be able to generate a working vm image and boot into it. Guix will be able to use the Hurd servers (i.e. /hurd/init) to start the init manager and the system itself. We will also have to account for the fact that servers in /hurd will be symlinks to /gnu/store/*, and modify Guix and/or the Hurd libraries accordingly to achieve functionality. Regarding the init daemon, please note that a precedent already exists for starting GuixSD/Hurd with GNU Shepherd as pid 1 (David Michael [2015]).

Finally, once GuixSD is successfully ported, we can cater to the last stage of taking advantage of Hurd specific mechanisms.

This includes but is not limited to:

- Replacing (gnu build linux-container) with (gnu build subhurd) when on a GNU/Hurd system. Subhurds can offer isolation similar to Linux containers as described here.
- Modify the guix-daemon to run without root privileges by utilizing the Hurd architecture. The daemon needs root privileges in order to setup chrooted environments for the builds. In the Hurd this can be done by setting up a translator as described here.
- Guix uses symlink trees for user profiles. Instead we can use stowfs. Stowfs creates a traditional Unix directory structure from all the files in the individual package directories.

3. Estimated Project Timeline

Before March 31

- Finish merging the wip-hurd branch to upstream.
- Write a (guix build hurd) module which will contain Guile bindings to the RPC stubs like hurd/fs.defs or hurd/exec.defs .

April 1 – April 15

- Package missing dependencies (Hurd libs).
- Re-implement ‘settrans’ in scheme.

April 16 – May 1

- At this point we will have the tools needed to build a Hurd based file-system. (Milestone 1)
- Start working on getting `(guix build *)` and `(gnu build *)` modules to work on Hurd.
- Make sure ‘%base-packages’ in `(gnu system)` module work as expected on Hurd.

May 2 - May 22

- Create `(gnu build hurd-boot)` and `(gnu build hurd-initrd)`.
- Start working on describing the GNU/Hurd system in (gnu system).

May 23 – 12 June

- Modify (gnu system *) modules as needed.
- All the modules (guix build *) and (gnu build *) will be working as expected by now.
- Try building a GuixSD image. (Milestone 2)

13 June – 23 June

- Solve any problems with booting into the system and running GNU Shepherd.

24 June – 9 July

- Have a fully working GNU/Hurd system. (Milestone 3)
- Make sure all the services/packages run correctly on the new GuixSD/Hurd system and solve any issues.

10 July – 8 August

- Start working on getting Hurd specific mechanisms integrated to Guix.

9 August - 23 August

- By now all the objectives will have been achieved.
- Merge patches, code refactoring, documentation.
- Deliver a working GuixSD system image with Hurd as the kernel.

4. About me

My name is Manolis Ragkousis and you can contact me via email or irc (‘phant0mas’). I am 22 years old and I am studying Informatics Engineering (Computer Science curriculum) in the Department of Informatics Engineering at the Technological Educational Institute of Crete. I am currently in the process of writing my Bachelor’s Thesis on running L4Linux and L4re/Fiasco-based realtime applications concurrently on a ZedBoard.

I am a Guix contributor since 2014. My main contribution to the project includes adding GNU/Hurd support to GNU Guix which was also part of GSoC 2015.

I am a supporter of Free Software and I want to help the GNU Guix and GNU Hurd projects in any way I possibly can. My belief is that my work could attract more people to get involved into both projects and help in increasing the number of developers. I will work on this project regardless of being accepted or not as I have kept doing for the last 2 years.

For a more detailed self-portrait, please refer to my online CV.

This project will be my only activity during the summer and I will work full time on it.