The System XVI Manual

Release manual-pre1.0

David Mackay

Mar 08, 2020

I: Introduction to System XVI

A forewarning: System XVI is not yet useable, not even in a limited fashion. It remains still prototypical, and will require significant work before making generally available: completion of basic features, making robust the code, and creation of tests. *The manual is made available to the public only under this proviso.*

This is the System XVI Manual. It aims to become an authoritative document on System XVI, or S16, the new System Resource Manager available for Linux and BSD.

CHAPTER 1

What is System XVI?

System XVI, abbreviated *S16*, is a modern *system resource manager* for Linux and the BSD distros. It manages system resources, ensuring that they are kept running, self-repairing after faults, and providing the administrator with the information necessary to make appropriate repairs if an un-self-reparable error should occur.

It is designed in line with the Four Motives, which are detailed below:

1.1 The Four Motives

- **Interface Orientation**: the system should be designed to fit a clean and stable interface. A well-designed interface makes for an obvious implementation.
- **Separation of Concerns**: individual components should not do much alone, but work in concert to create a grand system.
- **Modularity**: components should be easily replaceable and extensible.
- **Self-healing**: components that crash should be able to restart without forgetting system state or otherwise causing breakage.

1.2 Flexibility

System XVI is *generic*. It has been designed to allow for the management of many different kinds of system resources in a uniform way. For example, it can manage *background services* like *MongoDB* or *Node.js*, *chronological services* like a regular temporary-file cleaner, *internet services* like *vsftpd* or *telnetd*, as well as other kinds of services.

Because it follows the Four Motives, all this is achieved in separate daemons; each implements one section of the grand concert of System XVI. The flexibility allows, for example, legacy unit-files from SystemD to be converted and ran by System XVI.

1.3 Fault-Tolerant

System XVI is fault-tolerant. If a resource should fail, System XVI will try to make that resource available again, restarting services as needed to restore its functionality. In the case a severe fault occurs which cannot be repaired automatically, a clear description of the fault will be assembled and made available to the administrator for their viewing.

CHAPTER 2

Prior Art

Service management is not a particularly new subject. While *sysvinit* and *systemd* have garnered much of the attention span of the GNU/Linux world, with *upstart* having previously seen some interest, other competitors exist. These are described below:

2.1 IBM System Resource Controller

A simple system for running subsystems which are comprised of subservers (i.e. daemons.) Restarts subsystems automatically and provides notification of failure.

2.2 The Daemontools Family

daemontools is a service supervisor implemented by D.J. Bernstein in the late 90s, which is designed in line with Bernstein's principles of good software. These principles have led to a simple and elegant result. Since then, derivatives and reimplementations along similar lines have appeared. Some of these took on the question of broader-scale service/system management (as opposed to just supervision of daemons), e.g. dependency management. A few among the daemontools family are listed below:

- *runit* by Gerrit Pape: Designed from scratch to be useable as a replacement for *init* altogether. Used by the Void distro of GNU/Linux.
- *s6* by Laurent Bercot: Built with particular principles in mind, notably the ability to act as low-level components of a larger-scale service management suite. Accordingly *s6-rc* has been developed atop, which provides dependency management and the like. Includes a simple scripting language called *execline* for writing service start scripts in. A solution well-worth considering.
- *nosh* by Jonathan de Boyne Pollard. Like *s6-rc*, *nosh* provides higher-level service management features and is already equipped with a *systemd* unit-file converter and service descriptions for BSD and Linux. Named for its simple no-shell script interpreter. *nosh* is probably the best-designed cross-platform system/service manager.

2.3 OpenRC

Noted for its use on the Gentoo and Funtoo GNU/Linux distros. Lightweight, written in C and POSIX *sh*. Provides the traditional feel of init scripts while offering modern functionality.

2.4 Launchd

Apple's take on system and service management. Combines *init*, Mach's *mach_init*, *crond*, *atd*, *inetd*, and other low-level daemons into one. Previously open-source under the Apache License, now appears to have been closed (something to do with iPhone jailbreaking?), with the most recent code drop being some years old. This is likely to be the nail in the coffin for attempts to port launchd to FreeBSD.

2.5 SMF

The *Service Management Facility* of Illumos. System XVI is very similar in design to SMF.

CHAPTER 3

Principles of System XVI

System XVI aims to be a well-engineered system. It is a goal of paramount importance that the architecture of System XVI respect the Four Motives, which are described in *The Four Motives*.

CHAPTER 4

Workflow

System XVI is simple to build and work with. This section of the Developers' Handbook explains how to compile S16, make changes, and submit them for inclusion.

4.1 Dependencies

System XVI has few dependencies, but does require some in order to be built. Some of them are only neededed on particular platforms and others are totally optional. They are all detailed below:

4.1.1 Required: All Platforms

- readline-devel

4.1.2 Required: GNU/Linux

- procpsng-devel
- libkqueue-devel

4.1.3 Optional: All Platforms

- Java JDK 1.8 or later (for the S16 Java library)
- Python 3 (for the pre-commit checks)

4.2 Getting the Source Tree

The System XVI source tree is managed by the Git version control system. You will need it installed in order to check out the source code. If you wish to make changes to be submitted for integration, use GitHub to create a fork of the source tree and clone that. Otherwise, follow the instructions on the System XVI GitHub page to clone the repository.

Having cloned the repository, you must now set up the submodules. These are subtrees of the source tree which contain sources from other projects which S16 makes use of. Run this command to do so:

```
git submodule update --init --recursive
```

4.3 Building System XVI

System XVI leverages CMake for building. In order to build System XVI, you need to create a folder in which to build (for example, make a subfolder of the S16 tree called *build*), enter that directory, and then run:

```
cmake ..
```

After this, in order to build the sources, run:

```
make
```

This should run without any errors being generated. If any do appear, please make an Issue on the S16 GitHub page.

- genindex
- modindex
- search