s6

`s6` is a process supervision suite that also provides tools for service management (`s6-rc`) and system initialization/shutdown (`s6-linux-init`).

Contents

General overview

s6 software is all designed to be very modular and can be mixed and matched with other things. However, s6 software is also designed to work well with one another. In Artix, we take full advantage of what s6 offers and use all of the available tools to provide an /sbin/init, PID1, process supervisor suite, and service manager for users.

`s6-linux-init` is what actually initiates your system. It mounts a tmpfs onto `/run` and copies the `/etc/s6/current/run-image` directory into `/run`. `/sbin/init` then execs into the `s6-svscan` program (provided by `s6`) which functions as your PID1 for the lifetime of the machine. Once this is done, the stage 1 part of init is finished. In addition to being PID1, `s6-svscan` is also the root of your process supervision tree. It monitors every service found and appropriately spawns an `s6-supervise` for every service.

Additionally, during bootup the `rc.init` script in the `/etc/s6/current/scripts` directory gets executed. `s6` itself does not do service management, but only process supervision. In `rc.init`, the actual service manager, `s6-rc` is started. From that point, we launch the default runlevel defined by `s6-rc` and begin the desired services.

On shutdown, every service in s6-rc is brought down, followed by killing all `s6-supervise` processes and other processes, then unmounting everything, and finally killing `s6-svscan`.

Note: `s6-linux-init` on artix is compiled with the option to print messages to `/dev/console`. By default, this is tty1. You can change `/dev/console` to another location using a kernel parameter (see your bootloader's configuration) if you'd like.

Recovery: s6-linux-init always provides a getty service on tty12 that you can use for recovery purposes (in case s6-rc crashes, etc.). As long as the system correctly boots, it will be there.

Installation

Install the `s6-base` package.

Installation of services

s6 service packages are named *`package_name-s6`* and, when installed, will be available in `/etc/s6/sv`.

Programs

The s6 software suite comes with many different binaries, but in general you only need to directly interact with a small subset of them. Below are some of the more interesting programs.

- `s6-db-reload` - a helper script for updating and reloading the s6-rc database
- `s6-rc` - the main program for controlling and managing services
- `s6-rc-db` - a tool for analyzing a compiled service database
- `s6-svc` - a tool to directly send commands to an s6-supervise process
- `s6-svstat` - a tool for checking the current states of a process monitored by s6-supervise (an s6-rc longrun)

Files

Most of the files associated with the s6 software packages are installed in the `/etc/s6` directory.

- `/etc/s6/current` - the base directory for s6-linux-init
- `/etc/s6/current/scripts` - various startup/shutdown scripts executed by s6-linux-init
- `/etc/s6/config` - conf files for specific s6-rc services
- `/etc/s6/rc` - where compiled databases are stored; the current live database is always symlinked to `/etc/s6/rc/compiled`
- `/etc/s6/rc.local` - file for executing arbitrary shell commands on bootup (any shell scripts in /etc/local.d suffixed with *.start are executed on bootup and those with *.stop are executed on shutdown)
- `/etc/s6/skel` - contains the default startup/shutdown scripts that come with artix linux
- `/etc/s6/sv` - default directory for script packages from Artix
- `/etc/s6/adminsv` - directory for custom user services as well as script packages from Artix that allow for editing
- `/etc/s6/fallsv` - emergency fallback that is used if the s6-rc-compile of sv and

adminsv fails

Basic usage

A key concept to using s6-rc is to understand the notion of "bundles." You can take their namesake literally. A bundle in s6-rc is any collection of services, oneshots, and even other bundles. These are quite similar to openrc's runlevels and are used in similar ways in Artix. The package, `s6-scripts`, which contains essential startup oneshots and daemons for an Artix system internally uses many different bundles for convenience, but for users the main bundle they should concern themselves with is the `default` bundle. This is started by s6-rc and in general users will want to add their services to this bundle.

Note: There is a bundle within `default` called `boot`. This is a collection of startup/shutdown boot oneshots and daemons deemed essential for a working system. These are mostly provided by the `s6-scripts` package with some optional dependencies that install themselves into the appropriate directory.

Also note that s6-rc manages dependencies so it is not necessary to manually start all needed dependencies. When service `foo` is started, all of its dependencies (if they are not already up) are automatically started. Here are some handy commands.

- Stop a service/bundle `# s6-rc -d change service_name`
- Start a service/bundle `# s6-rc -u change service_name`
- Restart a service (only works with s6-rc longruns)
  `# s6-svc -r /run/service/service_name`
- List all active services `# s6-rc -a list`
- List all services/bundles in the database `# s6-rc-db list all`
- Check the status of an s6-rc longrun `# s6-svstat /run/service/service_name`

Updating bundle contents

Within every directory of a bundle, there is a `contents.d` folder which contains empty files named after services that are within the bundle. So in order to add services to a bundle, you just touch empty files named after the services in the directory. To add services to the default bundle (so it starts on boot), you would do:

```
# touch /etc/s6/adminsv/default/contents.d/service1
# touch /etc/s6/adminsv/default/contents.d/service2
# s6-db-reload
```

The `s6-db-reload` command is a symlink to the hook Artix uses to handle s6-rc database upgrades. It is executed whenever any *-s6 package in Artix is installed to ensure services are immediately available in the new database. This consists of 3 main steps. First, it compiles a new database with a unique name generated by the `date` command with `s6-rc-compile`. Then, it executes `s6-rc-update` to update the live database to the newly compiled database. Finally, it atomically updates the symlink to

`/etc/s6/rc/compiled` so on the next boot, the system executes the newest database.

Source directory structure

s6-rc has three types of services: longrun, oneshot, and bundle. Most *-s6 packages are longrun services (AKA daemons). Oneshot services do exactly what their name implies: execute once on bootup and optionally on shutdown. Bundles are simply a collection of longruns, oneshots, and even other bundles. Every source directory will have a mandatory `type` file that contains a single line defining what type the service is. Longrun services must contain a file called `run`, oneshots must contain a file called `up`, and bundles contain a subdirectory named `contents.d`.

In Artix, a typical service script comes with two different parts: the daemon itself and the logger daemon that uses the `s6-log` tool. The scripts for the actual service `foo` will be installed in the `/etc/s6/sv` directory as `foo-srv`. Additionally, a small logger daemon for that specific service will be installed in `foo-log`. `foo-log` will catch any output from `foo-srv` and save it in `/var/log/foo` directory. The `s6-log` logger daemon is run as the `s6log` user and group. To give a user permission to view logs, just add him to the `s6log` group. `s6-log` does log rotation for you and has many different configurable options.

When interacting with s6-rc, the name of `foo` doesn't change (i.e. you still do `# s6-rc -u change foo`). This starts both `foo-srv` and `foo-log` and handles any of the dependencies for you.

This is a tree of a longrun directory structure (aka `/etc/s6/sv/foo-srv`). In many scripts, only `producer-for`, `type`, and `run` exist.

```
foo-srv
├── dependencies.d (optional subdir containing names of
dependencies)
├── notification-fd (optional)
├── producer-for (required for foo-log)
├── run
└── type
```

A tree for a logger daemon longrun (aka `/etc/s6/sv/foo-log`) looks like this.

```
foo-log
├── consumer-for
├── notification-fd
├── pipeline-line
├── run
└── type
```

See also

- https://skarnet.org/software/s6/ - Official s6 documentation
- https://skarnet.org/software/s6-rc/ - Official s6-rc documentation
- https://skarnet.org/software/s6-linux-init/ - Official s6-linux-init documentation

---