runit

runit is a suite of tools which provides an init (PID 1) as well as daemontools-compatible process supervision framework, along with utilites which streamline creation and maintenance of services.

Contents

Installation

Install the `runit` package.

Installation of services

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

Programs

Runit has several programs, but usually you will only interact directly with one program only.

- `sv` - used for controlling services, getting status of services, and dependency checking.
- `chpst` - control of a process environment, including memory caps, limits on cores, data segments, environments, user/group privileges, and more.
- `runsv` - supervises a process, and optionally a log service for that process.
- `svlogd` - a simple but powerful logger, includes auto-rotation based on different

methods (time, size, etc), post-processing, pattern matching, and socket (remote logging) options.
- `runsvchdir` - changes service levels (runlevels, see below)
- `runsvdir` - starts a supervision tree
- `runit-init` - PID 1, does almost nothing besides being the init

Files

There are several files that will be installed by `runit`.

- `/etc/runit/1` - stage 1, system's one-time initialization tasks
- `/etc/runit/2` - stage 2, Normally runs `runsvdir`, should not return until the system is going to halt or reboot.
- `/etc/runit/3` - stage 3, system's shutdown tasks
- `/etc/runit/ctrlaltdel` - Runit will execute this when receiving a `SIGINT` signal
- `/etc/runit/runsvdir/*` - Runlevels
- `/etc/runit/sv/*` - directory containing subdirectories of available service files
- `/run/runit/service` - always symlinked to active runlevel, `sv` will search for running service here

However, since `runit` itself depends on `runit-rc`, there will be several extra rc files installed, most contained in `/etc/rc` and `/usr/lib/rc`.

Basic usage

Unlike other distros using runit, Artix doesn't store its service directory in `/var/service` or `/service`, but in `/run/runit/service` instead.

Service supervision

By default services provided by Artix packages aren't in the `svdir` and can't be started or managed by any tool. To do so it is needed to link the service directories into the desired runlevel directory. For example, the following command allows to supervise a service in the current runlevel:

```
ln -s /etc/runit/sv/service_name /run/runit/service
```

If the service isn't going to be used anymore, the previously created symbolic link has to be removed from the runlevel directory. For example, the following command disables supervision on a service in the current runlevel:

```
unlink /run/runit/service/service_name
```

Disable service

It can be needed to be able to manage a service but not to start it at boot time, to do so it is possible to disable it by adding an empty file named `down` into the service's

directory. Keep in mind that if a service has to be enabled in a runlevel but not in another runlevel, you have to duplicate the directory in `/etc/runit/sv`, to setup the two service directories differently and to manage symbolic links accordingly. For example the following command disables a service that is supervised in the current runlevel:

```
touch /run/runit/service/service_name/down
```

To enable the service again the file has to be removed. For example the following command enables a service supervised in the current runlevel:

```
unlink /run/runit/service/service_name/down
```

Managing services

Action	Command
Start	`sv up service_name` or `sv start service_name`
Stop	`sv down service_name` or `sv stop service_name`
Restart	`sv restart service_name`
Send SIGHUP	`sv hup service_name`
Check status	`sv status service_name`

Note that the two methods to start/stop services aren't equivalent and that other signals can be sent, read `sv(8)` for more details.

Runlevels

By default, runit has 2 runlevels, `default` and `single`. New runlevels can be created by creating a new folder in `/etc/runit/runsvdir/` and symlinking your desired service to that runlevel.

```
ln -s /etc/runit/sv/service_name /etc/runit/runsvdir/runlevel
```

To switch runlevel use the `runsvchdir` program. Note that when switching, all services are stopped then restarted if enabled in the new runlevel.

```
runsvchdir runlevel
```

Service directory structure

This is a tree of a complete service directory structure (aka `/etc/runit/sv/servicedir`), in some run scripts, typically only `run` will be available as usually it's the only file needed.

```
servicedir
├── run (755)
```

```
├── check (755)
├── conf (644)
├── finish (755)
└── log (directory)
    ├── config (644)
    └── run (755)
```

A runit (or any daemontools-compatible) service directory contains at least one executable file, named `run`, which runs process in the **foreground**.

This means that:

- The `run` file must be a symlink to the service executable, or a script that eventually `exec`'s into it.
- The service executable must not run in the background, otherwise runit will lose track of its state. If necessary, the `run` script should pass flags to the service executable that prevent this behavior (such as enabling debug mode).

The service directory may also contain executables like `finish` and `check`. `finish` will be executed when a service is stopped, and `check` will be executed (if exists) by `sv check` or `sv status`.

Some `run` scripts can source variables and additional commands from a user-supplied `conf` file (which should be created in the service directory, and is not executable). Usually, a `run` shell script that supports a `conf` file has the line `[ -r ./conf ] && . ./conf`.

If a service directory contains another directory named `log`, the output of the `run` process in the service directory will be piped to the input of the `run` process in the log directory. If the log service uses `svlogd`, it may be configured by using the file `config`. How `svlogd` can be configured is explained in the `svlogd(1)` manpage.

Service dependencies

Some services may depend on other services. For example, `NetworkManager` depends on `dbus`. To ensure that required dependencies are satisfied, check the service's `run` file. For example, for NetworkManager:

```
# /etc/runit/sv/NetworkManager/run
sv check dbus >/dev/null || exit 1
```

This means you have to enable `dbus` for NetworkManager to start.

Runit for user-services

To use runit to manage services at a user level, you will need to create a running runit user instance for your user. This instance will need it's own directory in which it can operate. It is recommended that this instance be managed by the system's runit

instance. The way this is setup is by creating a runit service with the following `run` file:

```
#!/bin/sh

export USER="your-username"
export HOME="/home/$USER"

groups="$(id -Gn "$USER" | tr ' ' ':')"
svdir="$HOME/.service"

exec chpst -u "$USER:$groups" runsvdir "$svdir"
```

The `svdir` can be changed to a directory of your liking and the `USER` must be changed to a user for which you want to setup user-services. Any new user service can be added to a configuration directory (i.e. `~/.config/sv`) from which they can be symlinked into the `svdir` directory (similarly to how `/etc/runit/sv/example-service` gets symlinked to `/run/runit/service`).

See also

- http://smarden.org/runit - Official runit documentation
- https://docs.voidlinux.org/config/services/index.html - Void Linux handbook on runit
- https://github.com/JojiiOfficial/rsv - rsv, an unofficial rewrite of the `sv` command in Rust with additional features
- https://github.com/madand/runit-services#user-services - user-services for runit

---