Runit

Introduction

runit is a UNIX init scheme with service supervision. It is a cross-platform Unix init scheme with service supervision, a replacement for sysvinit, and other init schemes and supervision that are used with the traditional init. runit is compatible with djb's daemontools.

In Unix-based computer operating systems, init (short for initialization) is the first process started during booting of the computer system. Init is a daemon process that continues running until the system is shut down. Slackware comes with its own legacy init (/sbin/init) from the sysvinit package, that used to be included in almost all other major Linux distributions.

The init daemon (or its replacement) is characterised by Process ID 1 (PID 1).

To read on the benefits of runit, see here: http://smarden.org/runit/benefits.html

* *Unless otherwise stated, all commands in this article are to be run by root.*

Use runit with traditional init (sysvinit)

runit is not provided by Slackware, but a SlackBuild is maintained on https://slackbuilds.org/. It does not have any dependencies.

As we do not want yet to replace init with runit, run the slackbuild with CONFIG=no:

```
CONFIG=no ./runit.SlackBuild
```

Then install the resulting package and proceed as follows:

```
mkdir /etc/runit/ /service/
cp -a /usr/doc/runit-*/etc/2 /etc/runit/
/sbin/runsvdir-start &
```

Starting via rc.local

For a typical Slackware-stlyle service, you can edit `/etc/rc.d/rc.local` file

```
if [ -x /sbin/runsvdir-start ]; then
  /sbin/runsvdir-start &
fi
```

and then edit write `/etc/rc.d/rc.local_shutdown`

```
#!/bin/sh
```

```
RUNIT=x$( /sbin/pidof runsvdir )
if [ "$RUNIT" !=  x ]; then
  kill $RUNIT
fi
```

Then give `rc.local_shutdown` executive permission:

```
chmod +x /etc/rc.d/rc.local_shutdown
```

and reboot

Starting via inittab (supervised)

Remove the entries in `/etc/rc.d/rc.local` and `/etc/rc.d/rc.local_shutdown` described above.

Edit `/etc/inittab`

```
cat >>/etc/inittab <<EOT
SV:123456:respawn:/sbin/runsvdir-start
EOT
```

and tell init to re-read its configuration, e.g.:

```
init q
```

or reboot

How to replace init with runit

runit provides `runit-init` which can be used to boot up the system.

If you followed the previous chapter, then stop runsvdir and reverse all the changes that you made on your system.

So, remove the last line from `/etc/inittab`

```
rm -rf /etc/runit /service
```

And reboot:

```
shutdown -r now
```

Also, it is best to re-build and re-install runit

This time, when running slackbuild, do pass not any value for CONFIG, or set it to yes:

```
unset CONFIG
```

```
sh runit.SlackBuild
```

and upgrade with the resulting runit package.

Disable services that are already provided by the traditional init:

```
cd /service/
rm *
```

Create a new TTY service for runit:

```
cd /etc/sv/
cp -ar agetty-tty1/ agetty-tty8/
sed -i 's/tty1/tty8/g' agetty-tty8/*
ln -s /etc/sv/agetty-tty8 /service
```

Start runit's stage 2 for testing:

```
/etc/runit/2 &
```

And check that the agetty is running, by logging in as an unprivileged user (not as root!)

```
CTRL+ALT+F8
```

If this has been successful, then reverse the last intervention:

```
kill $( pidof runsvdir )

rm /service/agetty-tty8
rm -rf /etc/sv/agetty-tty8
for N in {1..6}; do
  ln -s /etc/sv/agetty-tty${N} /service
done
```

Then, to boot in runit, reboot and enter `init=/sbin/runit-init` in the Lilo prompt. To do this, hit the tab key when you are presented with the Lilo menu as shown in Images 1 and 2 below.

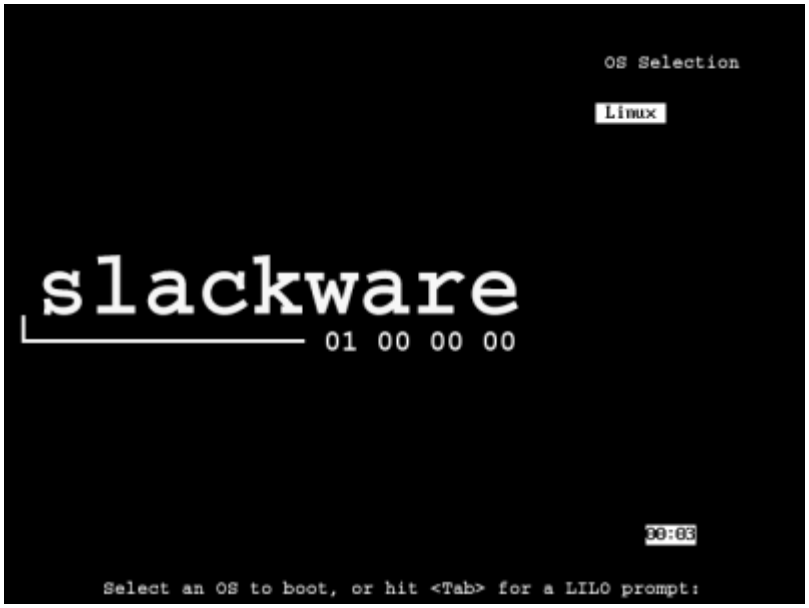

Image 1.: Hit the Tab key on the Lilo Menu before the counter times out.

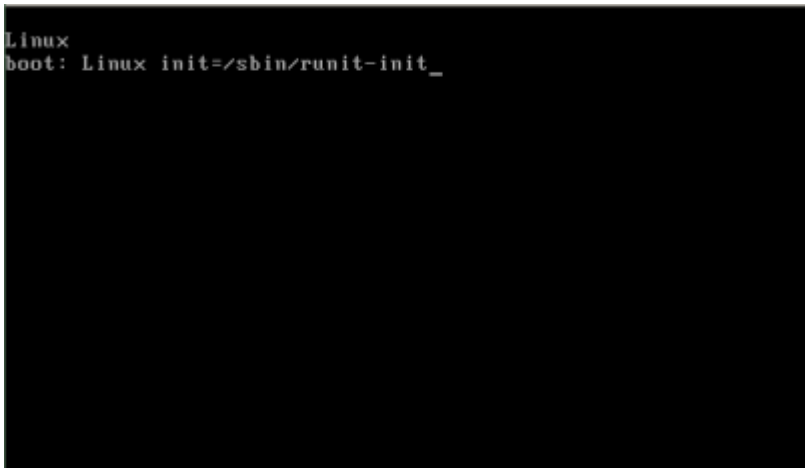

Image 2: Add init=/sbin/runit-init in the OS Selection Prompt

Confirm that runit has taken PID 1:

```
ps -o comm= -p 1
runit
```

If you do not have a multi boot system, you may append the init selection `in /etc/lilo.conf`

For example:

```
append="vt.default_utf8=0 init=/sbin/runit-init"
```

Then run lilo:

```
lilo -v
```

Or just replace `/sbin/init` with `/sbin/runit-init` like this:

```
cp /sbin/init /sbin/init.sysv
```

```
cp /sbin/runit-init /sbin/init
```

However, with the last alternative, be warned that if you had booted with the original init, the system may refuse to reboot, therefore boot from runit-init first.

Startup and Shutdown

The BSD style init scripts provided by Slackware are used to bring up the system.

Initially the single user script `/etc/rc.d/rc.S` runs. If set in the default runlevel 3, it follows it with `/etc/rc.d/rc.M`. In runit the initialisation script is located in `/etc/runit/1` for all runlevels. By default `/etc/runit/2` is restricted to the TTY services.

The script `/etc/rc.d/rc.6` is the rebooting script for the traditional init, while `/etc/rc.d/rc.0` (symlinked to the former) is the halting script. In runit, these scripts are called by `/etc/runit/3`

Note that the shutdown command provided by sysvinit package will not work in runit. To reboot you enter

```
/sbin/runit-init 6
```

and to halt enter

```
/sbin/runit-init 0
```

A convenient script is provided with the configuration files which is placed in `/usr/sbin/shutdown` that determines whether the system has booted with the traditional init or runit and then calls the relevant command accordingly.

The `halt`, `reboot` and `poweroff` commands provided by the sysvinit package would still work with runit.

Desktop Environments call `/sbin/shutdown` so they will not work until you move this file and symlink it to `/usr/sbin/shutdown`

```
mv /sbin/shutdown /sbin/shutdown.sysv
ln -s /usr/sbin/shutdown /sbin/shutdown
```

However consider that shutdown would not work for the original init.

runsv

Although runit may replace init as the PID 1 daemon, there is not much benefit unless other services are migrated from Slackware's BSD style to runsv(8). While this is not a complicated task, some familiarisation with runit's characteristics is recommended and makes the task easier. This short example should help illustrate runsv to new users.

Make an arbitrary directory under `/etc/sv/`

```
mkdir /etc/sv/example/
```

As a non-priviliged user (say user bob), create some directories:

```
mkdir -p /home/bob/runit/logs
```

Then edit a `/home/bob/runit/service.sh` script that simulates a (finicky) service in a convenient directory; say

```
#!/bin/bash
echo "Started service..."
for i in {1..10}
do
  echo "Doing stuff..."
  sleep 1
done
echo "Oh no I crashed..." >&2
exit 1
```

Give it executable permission:

```
chmod +x /home/bob/runit/service.sh
```

Run it to check and appreciate its execution:

```
/home/bob/runit/service.sh
```

Log in back as root and edit a file called `/etc/sv/example/run` and give it executable permissions:

```
#!/bin/sh -e
exec 2>&1
exec chpst -u bob /home/bob/runit/service.sh
```

```
chmod +x /etc/sv/example/run
```

Note the exec in the last line, it tells the shell that interprets the script to replace itself with the service daemon

Naturally edit as necessary.

Create another directory:

```
mkdir /etc/sv/example/log/
```

and edit another file under it, also called `/etc/sv/example/log/run`

```
#!/bin/sh
exec chpst -u bob svlogd -tt /home/bob/runit/logs
```

Give it executable permission:

```
chmod +x /etc/sv/example/log/run
```

Run the service to ensure that it works:

```
/etc/sv/example/run
```

Finally, we are ready to deploy the service. Create a symbolic link in `/service` to our staging location.

```
ln -s /etc/sv/example /service
```

Watch how your “service” works, crashes, but it is recovered by runit:

As your non-privilged user:

```
tail -f /home/bob/runit/logs/current
```

Hit CTRL+C to exit.

Finally, login back as root and let bob take the ownership of the whole `/etc/sv/example/` directory:

```
chown -R bob.users /etc/sv/example/
```

Now bob can manage this service with the `sv` command. Note that in most cases, services should belong to root, and only to root.

Managing Services with sv

To see the status of a supervised service use `sv s <service_name>`, for example,

```
sv s example
```

returns

```
run: example: (pid 42) 1587s
```

To see the status of all services, use

```
sv s /service/*
```

Stop/Start/Restart

Start a service

```
sv u example
```

Stop a service

```
sv d example
```

Restart a service

```
sv t example
```

Each of these is a shortcut, for 'up', 'down', and 'terminate', respectively. Only the first letter of each word is recognised.

More verbose forms of the above:

```
sv start example

sv stop example

sv restart example
```

Each of these will also return the status of the service upon exit.

Enabling a service

Service directories are placed under `/etc/sv/`. To enable a service in the current runlevel, create a symlink from it to `/service`.

```
ln -s /etc/sv/example /service/
```

Once a service is linked it will always start on boot and restart if it stops (unless this is disabled).

Disabling a service

To disable a service in the current runlevel remove the symlink to its service directory from `/service`.

```
rm /service/example
```

Removing the symlink will also stop the service.

Check out the sv man page to see additional options.

```
man sv
```

If you do not want a service to start upon bootup, or when enabled, just touch an empty file called down in the appropriate service directory.

```
touch /etc/sv/service/down
```

Dependencies

Dependencies of service are supported by starting the dependent run script as follows:

```
#!/bin/sh
sv start dependent-service || exit 1
.
.
exec ...
```

Runlevels

If you installed the slackbuild configuration files, you have two runlevels: default and single. The current runlevel is default. You can verify by looking under `/etc/runit/runsvdir/`, you will see that `/etc/runit/runsvdir/current` is a symbolic link `/etc/runit/runsvdir/default/`.

Change runlevel to another runlevel; single:

```
runsvchdir single
```

You will see that `/etc/runit/runsvdir/current` is now symlinked to `/etc/runit/runsvdir/single` and `/etc/runit/runsvdir/previous` is symlinked to `/etc/runsvdir/default`. If you reboot, you will boot again on the default runlevel as you have this entry in `/etc/runit/2`

```
runsvchdir default >/dev/null
```

You can create any other runlevel and name them as you like. To start, you can copy an existing runlevel directory and modify as you wish.

```
cp -ar /etc/runit/runsvdir/default /etc/runit/runsvdir/custom/
```

You can edit the runsvdir line of `/etc/runit/2` to your requirements.

Run scripts

Only one executable can be called for a service (the last line) and it must be called by the `exec` command. There are some generic runit run scripts on the Internet and on other Linux distributions. Also, other Slackers may post them on publicly available repositories. The author of this article has placed his here: https://gitlab.com/chrisabela/runit_scripts_for_slackware

Another set of installable run scripts are available at SBo.

In direct contrast to Slackware's BSD style service scripts, runit run scripts must not must run in the foreground, otherwise runit would think that they have crashed and restart them.

For some services, this may not be possible, but there are workarounds. You can forcefully terminate

them by ending the run script like this:

```
sv d service_name
```

Or you can use the `pause` command to keep the service alive. `pause` is a trivial command that will simply not exit, until it is killed (akin to tail -f /dev/null). A SlackBuild for `pause` is available at SBo. Then end the run script with:

```
exec chpst -b service_name pause
```

Migrating Services

It is suggested that services are migrated from stage 2 (which are still under the Slackware's BSD init scheme) to stage 3 carefully. Start from the bottom of `/etc/rc.d/rc.M` and work up. This means that you should start with any entries under `/etc/rc.d/rc.local`

Then continue for `/etc/rc.d/rc.S`

Proceed with other services to benefit's from runit features. Note that for some services, such as `atd` and `crond`, you would need to edit `/etc/rc.d/rc.M` as they are hard-coded.

Let's take the popular `NetworkManager` service as an example. According to the permission of `/etc/rc.d/rc.networkmanager` this script is called by `/etc/rc.d/rc.M` The latter is called by `/etc/runit/2` (stage 2).

Stop the service:

```
/etc/rc.d/rc.networkmanager stop
```

Disable it:

```
chmod -x /etc/rc.d/rc.networkmanager
```

Create a suitable directory for runit

```
mkdir /etc/sv/networkmanager/
```

Write a runit run script: `/etc/sv/networkmanager/run`

```
#!/bin/sh
prefix=/usr
exec_prefix=/usr
sbindir=${exec_prefix}/sbin
NETWORKMANAGER_BIN=${sbindir}/NetworkManager

export XDG_CACHE_HOME=/root/.cache
exec $NETWORKMANAGER_BIN -n > /dev/null 2>&1
```

Symlink the `/etc/sv/networkmanager` directory to `/service` to enable under current runlevel

and it will start in a few seconds:

```
ln -s /etc/sv/networkmanager /service
```

If run exits and `/etc/sv/<service name>/finish` exists (typically when the service is switched off), `runsv` runs `finish` if it has executable permission.

It is stuggested to let `udev` under runit's stage 2, but you can setup its monitoring for stage 3.

Using runit-init with other init scripts

OpenRC

Here `runit-init` is used for booting, which then transfers control to the OpenRC for things like mounting the filesystem, loading modules, running udev, etc.

It requires a working OpenRC system. Check the openrc page for instructions. The level 1 runit service uses the OpenRC `boot` and `sysinit` runlevels.

The gentoo wiki has more information, and runit-init-openrc details how to install and set it up.

void-runit

Here the scripts provided by the void-runit project are used to mount the filesystem, load modules, etc.

Currently this is the most independent way to setup runit.

Sources

1. http://smarden.org/runit/
2. https://en.wikipedia.org/wiki/Init
3. https://voidlinux.org/usage/runit/
4. https://www.youtube.com/watch?v=jiBIlhFxNJo
5. http://kchard.github.io/runit-quickstart/
6. https://www.slackbook.org/beta/
7. https://gitlab.com/chrisabela/runit_scripts_for_slackware
8. https://github.com/aadityabagga/runit-services
9. https://wiki.gentoo.org/wiki/Runit
10. https://github.com/void-linux/void-runit

- Written for Slackware 14.2 in December 2018
- Originally written by Chris Abela

init, runit