runit and ignite: a suckless init system?

Christian Neukirchen
chneukirchen@gmail.com

slcon 2013 · 22jun2013
Motivation

Since mid-2012, there has been a strong drive by many Linux distributions towards replacing old sysvinit with newer systems such as upstart (Ubuntu), openRC (Gentoo) and systemd (everything else).

While these implement some long wished-for features, they also add lots of complexity, often in the wrong places.

When Arch Linux decided to move towards systemd, I had to pull the communication cord...
What *init*(8) must do:

- Be pid 1.
- Reap children (handle SIGCHLD).
- Run something to bring up the rest of the system.
- Don’t crash. Ever.

Everything else probably does not belong into pid 1.
What an *init system* should do:

- Configure the system according to some files or boot flags.
- Allow to start and stop services.
Service supervision

When we want a service to run, it should run.

When we want a service to stop, it should be stopped.

How can we know a service is running? sysvinit (and badly written systemd descriptions) use *pid files*, and check if the pid is still running, or kill it.

**This is inherently racy.** The pid can get invalidated and reassigned at any time.
Service supervision

The *only* way to reliably run a service is to be its parent process. (You then get the pid from *fork(2)*.)

This also means *daemonization* is a waste of time.
runsv(8)

runit provides runsv(8), which starts and monitors a single service.

runsv switches to the directory service and starts ./run. If ./run exits and ./finish exists, runsv starts ./finish. If ./finish doesn’t exist or ./finish exits, runsv restarts ./run.

If ./run or ./finish exit immediately, runsv waits a second before starting ./finish or restarting ./run.
**sv(8)**

`runit` provides a tool `sv(8)` to talk to a running `runsv`:

- `sv status`
- `sv up`
- `sv down`
- `sv pause/cont/hup/alarm/interrupt/quit/1/2/term/kill`
- `sv exit`
- `sv restart`
- `sv check` (can be used to implement dependencies)

These tools communicate over a FIFO `.supervise/control` in the service directory.
runsvdir(8)

runsvdir(8) starts a runsv(8) process for each subdirectory, or symlink to a directory, in the services directory dir, up to a limit of 1000 subdirectories, and restarts a runsv(8) process if it terminates.
Putting it all together

We now have everything we need for service supervision.

You can use the mentioned parts on almost every Unix system already. (It runs on Linux, *BSD, MacOS X and Solaris.) I highly recommend it!

But we wanted to replace *init*(1), too.
runit(8)

- runit(8) must be run as pid 1.
- runit runs /etc/runit/1 and waits for it to terminate.
- runit runs /etc/runit/2, which should not return until system shutdown.
- If runit is told to shutdown the system, or stage 2 returns, it terminates stage 2 if it is running, and runs /etc/runit/3.
- If stage 3 returns, runit checks if the file /etc/runit/reboot exists and has the execute by owner permission set. If so, the system is rebooted, it’s halted otherwise.
- If runit receives a CONT signal and the file /etc/runit/stopit exists and has the execute by owner permission set, runit is told to shutdown the system.
The tools are there now.

But we don’t have the `/etc/runit/\{1,2,3\}` scripts yet to actually do the stuff. (Nor do we have all the service directories.)

This is the task fulfilled by the *ignite* project.
ignite

*ignite* is a set of shell scripts to boot an Arch installation with *runit*.

It includes `/etc/runit/{1,2,3}` as well as service directories for many popular services (53 currently).

`/etc/runit/{1,3}` were written after a close reading of Arch *initscripts-2012.05.1*, back when these were simple bash scripts.

They support the `rc.conf` configuration file for system-wide configuration that Arch used to have.
These scripts are pretty straight forward:

```bash
% wc -l /etc/runit/?
  83 /etc/runit/1
  40 /etc/runit/2
  46 /etc/runit/3
169 total
```
How to boot Arch Linux? · /etc/runit/1

- Mount /proc, /sys, /run, /dev, /dev/pts, /dev/shm
- Remount / read-only
- Enable Unicode for all Linux consoles, and load console fonts and fontmaps
- Load the console keymap
- Get the clock from the hardware clock
- Run udev
- Turn on local networking (interface lo)
- Set the hostname
- Deal with dmraid, btrfs, LVM, cryptsetup
- fsck if needed or requested
- Remount / read-write
- Mount other filesystems
- Enable swap
- Set the time zone
- Seed the random number generator
- Clear some files
- Save boot dmesg
What next? · /etc/runit/2

- Assemble list of services we want to run from rc.conf.
- Run /etc/rc.local, for one-time jobs (e.g. set some power saving options).
- Exec runsvdir(8).
Shutting down Arch Linux? · /etc/runit/3

- Stop all services
- Run /etc/rc.local.shutdown
- Save the random number generator seed
- Quit udev
- Kill all processes still around
- Turn off swap
- Unmount everything but /
- Close all cryptdevices, LVM
- Remount / read-only
- Tell runit to halt or reboot
Sample services

#!/bin/sh
install -d -m 0755 -o root -g root /var/run/dovecot
exec dovecot -F

We include a `pause(2)` utility if there is nothing to supervise:

#!/bin/sh
/usr/sbin/alsactl restore
exec chpst -b alsa pause

Most services are easy to write.
ignite in practice

runit
- runsvdir -P /run/runit/runsvdir/current...
  - runsv dovecot
    - dovecot -F
      - anvil
      - imap
    - log
  - runsv postfix
    - master -d
      - pickup -l -t unix -d -u
      - qmgr -l -t unix -d -u
    - tlsmgr -l -t unix -d -u
  - runsv sshd
    - sshd -D
  - runsv ntpd
```
-ntpd -g -u ntp -n
-runsv crond
  -crond -n
    -crond -n
      -(run-parts)
-runsv wlan0-wpa
  |-logger -t wpa_supplicant-
    -wpa_supplicant -i wlan0 -D nl80211,wext...
-runsv wlan0
  `-dhcpcd -qLB -t 0 wlan0
-runsv eth0
  `-dhcpcd -qLB -t 0 eth0
-runsv syslog-ng
  `-syslog-ng -F
  `-runsv agetty-tty1
    `-agetty -8 -s 38400 --noclear tty1 linux
  `-udevd --daemon
```
ignite features: general

- Support for old-style rc.conf (snippet from my notebook):

  TIMEZONE="Europe/Berlin"
  MODULES=(acpi_cpufreq coretemp pcrypt snd-pcm-oss
           hdaps tp_smapi kvm-intel)
  DAEMONS=(agetty-tty{1,2,3,4,5,6} syslog-ng smartd dkms
           hdapsd alsa !laptop-mode eth0 wlan0 wlan0-wpa crond
           ntpd acpid sshhd postfix dovecot cpupower dbus mpd
           unbound batt-led)

- Services for networking: DHCP and static configurations
- Pretty straightforward to write service scripts for most dæmons
- sysvinit feelalikes for halt(8), reboot(8), shutdown(8).
ignite features: robustness

- Interrupt support during boot: drop into a rescue shell in case boot fails
- Single-user mode support with read-only / (amazingly hard these days)
- Boot logging into dmesg kernel buffer

```
[ 34.630993] :: mount -o remount,rw /
[ 34.631160] EXT4-fs (dm-2): re-mounted. Opts: (null)
[ 34.633807] :: mount -a -t "nosysfs,nonfs,nonfs4,nosmbfs,nocifs" -o no_netdev
[ 34.635186] EXT4-fs (sdb1): mounting ext2 file system using the ext4 subsystem
```
ignite features: boot time

- Faster than *sysvinit*, since *runsvdir* starts all services in parallel.
- Slightly slower than *systemd* in practice:
  - need to wait for *udev* to settle completely
  - enabling UTF-8 on all 64 consoles takes some time in shell
  - *runsv* waits a second before it starts the service (could be patched out)
- In general, I don’t think boot time is that important: *It’s fast enough.*
ignite features: resource usage

- Small and mature code base (*runit* with all tools is about 5kLOC):

  ```
  % wc -l runit.c sv.c runsv.c runsvdir.c
  346 runit.c
  387 sv.c
  607 runsv.c
  286 runsvdir.c
  ```

- Very lightweight, thanks to statically linked musl binaries (x86_64):

<table>
<thead>
<tr>
<th>text</th>
<th>data</th>
<th>bss</th>
<th>dec</th>
<th>hex</th>
<th>filename</th>
</tr>
</thead>
<tbody>
<tr>
<td>12320</td>
<td>224</td>
<td>424</td>
<td>12968</td>
<td>32a8</td>
<td>/sbin/runit</td>
</tr>
<tr>
<td>20044</td>
<td>224</td>
<td>888</td>
<td>21156</td>
<td>52a4</td>
<td>/sbin/runsv</td>
</tr>
<tr>
<td>17460</td>
<td>280</td>
<td>8904</td>
<td>26644</td>
<td>6814</td>
<td>/sbin/sv</td>
</tr>
<tr>
<td>32080</td>
<td>2156</td>
<td>672</td>
<td>34908</td>
<td>885c</td>
<td>/sbin/init.sysv</td>
</tr>
<tr>
<td>881017</td>
<td>106924</td>
<td>2497</td>
<td>990438</td>
<td>f1ce6</td>
<td>/usr/lib/.../systemd</td>
</tr>
</tbody>
</table>
• Very low overhead (measured on a Raspberry Pi):

<table>
<thead>
<tr>
<th>PID</th>
<th>TIME</th>
<th>MAJFL</th>
<th>TRS</th>
<th>DRS</th>
<th>RSS</th>
<th>%MEM</th>
<th>COMMAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0:02</td>
<td>33</td>
<td>735</td>
<td>3980</td>
<td>2488</td>
<td>1.1</td>
<td>/sbin/init</td>
</tr>
<tr>
<td>66</td>
<td>0:00</td>
<td>4</td>
<td>165</td>
<td>3294</td>
<td>1072</td>
<td>0.5</td>
<td>.../systemd-journald</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PID</th>
<th>TIME</th>
<th>MAJFL</th>
<th>TRS</th>
<th>DRS</th>
<th>RSS</th>
<th>%MEM</th>
<th>COMMAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0:01</td>
<td>0</td>
<td>13</td>
<td>142</td>
<td>20</td>
<td>0.0</td>
<td>runit</td>
</tr>
<tr>
<td>253</td>
<td>0:00</td>
<td>2</td>
<td>21</td>
<td>162</td>
<td>40</td>
<td>0.0</td>
<td>runsvdir</td>
</tr>
<tr>
<td>267</td>
<td>0:00</td>
<td>0</td>
<td>21</td>
<td>142</td>
<td>32</td>
<td>0.0</td>
<td>runsv sshd</td>
</tr>
<tr>
<td>269</td>
<td>0:00</td>
<td>1</td>
<td>21</td>
<td>142</td>
<td>32</td>
<td>0.0</td>
<td>runsv crond</td>
</tr>
<tr>
<td>270</td>
<td>0:00</td>
<td>0</td>
<td>21</td>
<td>142</td>
<td>32</td>
<td>0.0</td>
<td>runsv eth0</td>
</tr>
<tr>
<td>272</td>
<td>0:00</td>
<td>0</td>
<td>21</td>
<td>142</td>
<td>32</td>
<td>0.0</td>
<td>runsv agetty-tty1</td>
</tr>
</tbody>
</table>

• Runs $here on about a dozen different machines (i686, x86_64, arm) for various tasks (notebooks, desktops, wall-hung tablet, print servers, NAS) with very similar setup.

• Generally stays out of your way.
Problems

• Some programs cannot daemonize (or make it tricky)
• Not all services are implemented (NFS)
• Some dependencies are hard/too general to describe
• Supervision of very early tasks
  • udev is not supervised (yet?)
  • want to run ntpdate early, but need network up already...
• Still using syslog-ng
  • works, but complicates proper logging of shutdown
  • the runit way is to use svlogd(8) (much work to adapt all services)
• Arch defaults converge to systemd, not always reasonable (increases setup cost)
Summary

*runit* provides a lightweight, flexible and high-quality init system.

*ignite* shows that common Linux distributions can adopt *runit* without too much effort.

I recommend using *runit* for a suckless Linux distribution.
How to get it

- http://smarden.org/runit/
- http://github.com/chneukirchen/ignite
- #ignite on irc.freenode.net
- packer –S ignite-git (read the manual first)
Questions?