OpenBSD rc.d(8)

EuroBSDcon 2016
24 September, 2016

Antoine Jacoutot <ajacoutot@openbsd.org>
● OpenBSD developer since 2006
● ajacoutot@ aka aja@
● sysmerge, rc.d, rcctl, libtool, stuff, other stuff…
● >400 ports, GNOME (Foundation member)
● ftp.fr.openbsd.org
rc.d(8) was brought to you by

Robert Nagy <robert@openbsd.org>
Ingo Schwarze <schwarze@openbsd.org>
Antoine Jacoutot <ajacoutot@openbsd.org>
Stuff we're going to talk about

- historical (& current) system boot process
- rc.d alternatives and requirements
- rc.d usage
- rc.subr internals
- rcctl
KEEP CALM AND USE SHELL
I can has consistency?

- kill -HUP
- apachectl graceful
- rndc reload
- haproxy -sf $(cat /var/run/haproxy.pid)
The 90's called...

- boot loader -> kernel -> init
- init(1) uses sh(1) to run /etc/rc
- dependable, predictive, sequential
- dependency-less
Controlling the startup

/etc/rc.conf, default configuration
/etc/rc.conf.local, rc.conf(8) overrides

daemon_flags=flags|NO
service=YES|NO
rc.d requirements

- current paradigm cannot change
- preserve existing behavior
- plug rc.d on top (!= replacement)
- only handle daemons
- small, simple, robust, comprehensive
- easily debuggable
Alternatives at the time

- SMF, launchd
- OpenRC
- runit, daemontools
- Slackware Linux rc.d
- FreeBSD and NetBSD rc.d + rcorder
- ...
- small and targeted to our requirements
- no supervision
- no event driven / socket activated
- no parallelization
- no automatic startup ordering
October 2010: first implementation

/etc/rc.d/rc.subr, /etc/rc.d/foobar

designed for ports only

base was the ultimate goal
Initial implementation

- standard facility to signal daemons: `kill(1)`
- does not rely on PID files
- no `start-stop-daemon(8)`...
- good enough for ~95% of the ecosystem
- shell (ksh)
Initial implementation

- rc.d scripts initially called from /etc/rc.local
  - no disruption to the existent
  - traditional way to start external daemons
  - naming
    - same name as the daemon it is referring to (some exceptions)
    - dash -> underscore
      (script used as a var by the framework)
for _r in $rc_scripts; do
    [ -x /etc/rc.d/${_r} ] && \
    /etc/rc.d/${_r} start && \
    echo -n " ${_r}"
done
- sourced by rc.d scripts
- provides all subroutines
- 54 LOC at that time
“Who would need such a bloated interface?”
Aaaannd… we're in!

- one release later: base system daemons
- why the change of mind?
  - process not started in isolation
  - unexpected and/or dangerous behavior

"su(1) -l" for environment sanitation
su root -c 'apachectl2 start'

versus

su root -c '/etc/rc.d/apache2 start'
<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUTHORITY</td>
<td>/var/run/gdm/auth-for-ajacoutot-m3vPl9/database</td>
</tr>
<tr>
<td>EC2 HOME</td>
<td>/usr/local/ec2-api-tools</td>
</tr>
<tr>
<td>LOGNAME</td>
<td>ajacoutot</td>
</tr>
<tr>
<td>WINDOWID</td>
<td>39950112</td>
</tr>
<tr>
<td>LC_PAPER</td>
<td>en_US.UTF-8</td>
</tr>
<tr>
<td>HOME</td>
<td>/root</td>
</tr>
<tr>
<td>JAVA_HOME</td>
<td>/usr/local/jdk-1.7.0</td>
</tr>
<tr>
<td>MORE</td>
<td>-e</td>
</tr>
<tr>
<td>GDM_LANG</td>
<td>en_US.UTF-8</td>
</tr>
<tr>
<td>XMODIFIERS</td>
<td>@im=ibus</td>
</tr>
<tr>
<td>LC_MONETARY</td>
<td>en_US.UTF-8</td>
</tr>
<tr>
<td>GNOME_DESKTOP_SESSION_ID</td>
<td>this-is-deprecated</td>
</tr>
<tr>
<td>XDG_SESSION_COOKIE</td>
<td>peck.home.bsdfrog.org-1457525880.169095-987613489</td>
</tr>
<tr>
<td>LANG</td>
<td>en_US.UTF-8</td>
</tr>
<tr>
<td>SSH_AUTH_SOCK</td>
<td>/tmp/ssh-fY14JcellEs/agent.20253</td>
</tr>
<tr>
<td>LC_MEASUREMENT</td>
<td>en_US.UTF-8</td>
</tr>
<tr>
<td>SHELL</td>
<td>/bin/ksh</td>
</tr>
<tr>
<td>TERM</td>
<td>xterm-256color</td>
</tr>
<tr>
<td>DBUS_SESSION_BUS_ADDRESS</td>
<td>unix:path=/tmp/dbus-bTXFGN5XVm.guid=c1ba1bc5f3988d9ee7337f4156e0147b</td>
</tr>
<tr>
<td>USERNAME</td>
<td>ajacoutot</td>
</tr>
<tr>
<td>LC_NUMERIC</td>
<td>en_US.UTF-8</td>
</tr>
<tr>
<td>XDG_MENU_PREFIX</td>
<td>gnome-</td>
</tr>
<tr>
<td>WINDOWPATH</td>
<td>5</td>
</tr>
<tr>
<td>XDG_SESSION_TYPE</td>
<td>x11</td>
</tr>
<tr>
<td>PWD</td>
<td>/home/ajacoutot</td>
</tr>
<tr>
<td>DESKTOP_AUTOSTART_ID</td>
<td>10577b4c3ea13dc5f41457525883346266000000287180001</td>
</tr>
<tr>
<td>PKG_PATH</td>
<td>ftp.fr.openbsd.org</td>
</tr>
<tr>
<td>LD_LIBRARY_PATH</td>
<td>/usr/local/lib</td>
</tr>
<tr>
<td>LC_CTYPE</td>
<td>en_US.UTF-8</td>
</tr>
<tr>
<td>DISPLAY</td>
<td>:0</td>
</tr>
<tr>
<td>SSH_AGENT_PID</td>
<td>16845</td>
</tr>
</tbody>
</table>

“Too much information!”
OpenBSD startup sequence

- do things -> start_daemon() -> do other things -> start_daemon() -> ...
- hostname.if, rc.securelevel, rc.local, rc.shutdown
- run_upgrade_script() (sysmerge, firsttime)

rc.d = small subset of the startup sequence
- rc.subr 224 LOC
- /etc/rc -150 LOC
  - source rc.subr (functions only)
  - start_daemon()
  - start/stop pkg_scripts (while loop)
- big feature gain for 70 LOC
4+1 actions available

- `start` the daemon (flags, timeout, user, class, rtable)
- `stop` the daemon (SIGTERM)
- `reload` the daemon (SIGHUP)
- `check` if the daemon is running (pgrep)
- `restart` the daemon (stop && start)
- need to run as a privileged user (~!check)
- fully configurable and overridable
- main user interface: just a few knobs
#! /bin/sh
#
# $OpenBSD$

daemon="/path/to/daemon"

. /etc/rc.d/rc.subr

rc_cmd $1
2 optional flags

- -d debug mode
  - describe and display stdout/stderr
- -f force mode
  - similar to onestart
  - no-op for packages rc.d scripts
Enabling daemons

- `daemon_flags`
  - base system daemons

- `pkg_scripts` (ordered or reversed)
  - package daemons
- `daemon_class`
  - default: daemon
  - BSD login class the daemon will run under
    (resource limits, environment variables... )
- `daemon_flags`
  - default: `NO|<empty>` (from `/etc/rc.conf`)
  - flags passed to the daemon
- daemon_rtable
  - default: 0
  - routing table to run the daemon under
rc.d variables

- daemon_timeout
  - default: 30
  - maximum time in seconds to start/stop/reload
• daemon_user
  ○ default: root
  ○ user the daemon will run as
rc.d variables

- Variables are overridable by:
  - the rc.d script itself
  - /etc/rc.conf
  - /etc/rc.conf.local
rc.d variables

- /etc/rc.d/netsnmpd
  - `daemon_flags="-u _netsnmp -I -ipv6"

- rc.conf.local
  - `netsnmpd_flags=-u _netsnmp -a`

**override**: rc.d script name is substituted to *daemon* in the variable name
set to a login class of the same name as the rc.d script

- `netsnmpd_class=myclass`
apmd_flags=-A
hotplugd_flags=
saned_flags=-s128
ntpd_flags=NO
pkg_scripts=messagebus saned cupsd
• meta rc.d script
  ○ /etc/rc.d/samba start
  ○ /etc/rc.d/smdb start & & \n      /etc/rc.d/nmbd start
- multiple instances of the same daemon
  - `ln -s /etc/rc.d/foobar /etc/rc.d/foobar2`
  - `pgrep(1)` much match the correct one!
  - `foobar2_flags, foobar2_user...`
- entry point
- where the whole framework is defined
- sourced by rc.d scripts
  - to get std functions and default vars
  - functions can be overridden by the script itself
${rcexec} "${daemon} ${daemon_flags} ${_bg}"

cexec="su -l -c ${daemon_class} -s /bin/sh ${daemon_user} -c"
[ "${daemon_rtable}" -eq 0 ] || |
    cexec="route -T ${daemon_rtable} exec ${rcexec}"

cb_g=YES -> "&"

e.g.
  su -l -c daemon -s /bin/sh root -c "/usr/sbin/sshd -flags"
pkill -T "${daemon_rtable}" -xf "${pexp}"

pexp="${daemon}${daemon_flags:+ ${daemon_flags}}"

At shutdown: base system daemons scripts are **not** run (SIGTERM)
pkill -HUP -T "${daemon_rtable}" \
-xf "${pexp}"
pgrep -T "${daemon_rtable}" -q -xf "${pexp}"
Optional function: rc_pre()

- `start` will invoke `rc_pre()` before starting a daemon
- pre-launch time requirements
  - e.g. create a directory to store a socket
invoked by *stop* after a daemon process has been killed

- cleanup
  - remove dangling lock files
  - putting the system back into a pristine state (e.g. cups)
- main function
- last command called by an rc.d script
- 1 of 5 arguments
• check that the daemon is enabled
• check it is not already running
• run rc_pre()
• run rc_start()
• daemon variables in /var/run/rc.d/${rcscriptname}
• wait up to ${daemon_timeout} seconds
- check that the daemon is running
- run rc_stop()
- wait up to ${daemon_timeout} seconds
- run rc_post()
- rm /var/run/rc.d/${rcscriptname}
rc_cmd() restart

- /etc/rc.d/daemon stop
- /etc/rc.d/daemon start
- check that the daemon is running
- run rc_reload()
● rc_check()
some daemons do not support an action
  - turn function into a variable set to “NO”
    - e.g. rc_reload=NO
The `rc_usercheck` variable

- if `rc_check()` requires higher privileges
  - `rc_usercheck=NO`
match currently running process in case configuration changed

- e.g. /var/run/rc.d/ntpd

  - daemon_class=daemon
  - daemon_flags=-s
  - daemon_rtable=0
  - daemon_timeout=30
  - daemon_user=root
  - pexp=/usr/sbin/ntpd -s
daemon="/path/to/bin/foobar --daemonize"

#daemon_flags=
#daemon_rtable="0"
#daemon_timeout="30"
#daemon_user="root"

./etc/rc.d/rc.subr

#pexp="${daemon}${daemon_flags:+ ${daemon_flags}}"
#rc_bg=
#rc_reload=
#rc_usercheck=YES
#rc_pre() { }
#rc_start() { ${rcexec} "${daemon} ${daemon_flags} {_bg}" } 
#rc_check() { pgrep -T "${daemon_rtable}" -q -xf "${pexp}" } 
#rc_reload() { pkill -HUP -T "${daemon_rtable}" -xf "${pexp}" } 
#rc_stop() { pkill -T "${daemon_rtable}" -xf "${pexp}" } 
#rc_post() { }

rc_cmd $1
- rc.conf.local "editor" (sorting)
- configure & control daemons and services
- ala service(8) + chkconfig(8) + sysconfig
- syntax not compatible with service(8)
- alternative, not an $EDITOR replacement
multicast=YES

sshd=YES

multicast=

sshd_flags=

multicast_flags=NO

sshd_flags=NO
- unified interface
- abstraction
- daemon versus service
- regular versus meta script
- rcctl support in Puppet, Ansible and Salt
  - puppet: 120 additions and 441 deletions
rcctl -> rc.subr -> rc.d script -> rc.conf+rc.conf.local

- rc.subr

- **FUNCS ONLY=1**

- from sourced to parsed: |_rc_parse_conf()|

- stop injecting shell code in dangerous places
usage:  rcctl get|getdef|set service | daemon [variable [args]]
rcctl [-df] start|stop|restart|reload|check daemon ...
rcctl disable|enable|order [daemon ...]
rcctl ls all|failed|off|on|started|stopped
rcctl enable multicast messagebus cupsd
rcctl set ntpd flags -s
rcctl get all
rcctl restart ntpd smtpd sshd
rcctl ls started

“rcctl ls failed” is run daily(8)
● ! replacement for the traditional BSD init
● ! process control framework
● ! service supervisor
● compromise
  ○ may not be suitable for all possible uses
- boringly simple and robust
- preserved the original paradigm
- built on decades-old components
- consistent and unified interface with rcctl
- easy integration into other OSes
Questions?

Thank you EuroBSDcon!

Antoine Jacoutot
<ajacoutot@openbsd.org>
The OpenBSD Project