OpenBSD rc.d(8)

AsiaBSDCon

March 13th 2016

Antoine Jacoutot <ajacoutot@openbsd.org>

whoami(1)

- OpenBSD developer since 2006
- ajacoutot@ aka aja@
- cloud and automation consultant at D2SI

whoami(1)

- sysmerge, rc.d, rc.subr, rcctl, libtool…
- >450 ports, GNOME (Foundation member)
- ftp.fr, exopi

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

“I went to Japan and I all I got to see was a talk about a shell script!”

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

Initial landing

- 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)
- PID files are bad
- ~95% is good enough
- no start-stop-daemon(8)
- shell (ksh)

Initial implementation

- rc.d scripts initially called from /etc/rc.local
  - no disruption to the existent
  - traditional way to start external daemons

/etc/rc.local

```
for _r in $rc_scripts; do
   [ -x /etc/rc.d/${_r} ] && \
        /etc/rc.d/${_r} start && \
        echo -n " ${_r}"
done
```

/etc/rc.d/rc.subr

- sourced by rc.d scripts
- provides all subroutines
- 54 loc at that time

“Who would need such a bloated interface?”

We're in!

- 1 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

Environment leakage

```
su root -c 'apachectl2 start'
```

versus

```
su root -c '/etc/rc.d/apache2 start'
```

XAUTHORITY	/var/run/gdm/auth-for-ajacoutot-m3vPl9/database
EC2_HOME	/usr/local/ec2-api-tools
LOGNAME	ajacoutot
WINDOWID	39950112
LC_PAPER	en_US.UTF-8
HOME	/root
JAVA_HOME	/usr/local/jdk-1.7.0
MORE	-e
GDM_LANG	en_US.UTF-8
XMODIFIERS	@im=ibus
LC_MONETARY	en_US.UTF-8
GNOME_DESKTOP_SESSION_ID	this-is-deprecated
XDG_SESSION_COOKIE	peck.home.bsdfrog.org-1457525880.169095-987613489
LANG	en_US.UTF-8
SSH_AUTH_SOCK	/tmp/ssh-fVY14JcellEs/agent.20253
LC_MEASUREMENT	en_US.UTF-8
SHELL	/bin/ksh
TERM	xterm-256color
DBUS_SESSION_BUS_ADDRESS	unix:path=/tmp/dbus-bTXFGN5XVm,guid=c1ba1bc5f3988d9ee7337f4156e0147b
USERNAME	ajacoutot
LC_NUMERIC	en_US.UTF-8
XDG_MENU_PREFIX	gnome-
WINDOWPATH	5
XDG_SESSION_TYPE	x11
PWD	/home/ajacoutot
DESKTOP_AUTOSTART_ID	10577b4c3ea13dc5f14575258833462660000287180001
PKG_PATH	ftp.fr.openbsd.org
LD_LIBRARY_PATH	/usr/local/lib
LC_CTYPE	en_US.UTF-8
DISPLAY	:0
SSH_AGENT_PID	16845

“Too much information!”

OpenBSD startup sequence

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

rc.d = small subset of the startup sequence

rc.d today

- rc.subr +219 loc
- /etc/rc -150 loc
- big feature gain for 70 loc

Features and usage

- 4+1 actions available
  - *start* the daemon (flags, timeout, user, class)
  - *stop* the daemon (SIGTERM)
  - *reload* the daemon (SIGHUP)
  - *check* if the daemon is running (pgrep)
  - *restart* the daemon (stop && start)

Actions

- need to run as a privileged user (~!check)
- fully configurable and overridable
- main user interface: just a few knobs

Minimal rc.d script

```
#!/bin/sh
#
# $OpenBSD$

daemon="/path/to/daemon"
. /etc/rc.d/rc.subr
rc_cmd $1
```

Actions

- 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)
  - package daemons

rc.d variables

- daemon_class
  - default: daemon
  - BSD login class the daemon will run under (resource limits, environment variables...)

rc.d variables

- daemon_flags
  - default: NO|<empty> (from /etc/rc.conf)
  - flags passed to the daemon

rc.d variables

- daemon_timeout
  - default: 30
  - maximum time in seconds to start/stop/reload a daemon

rc.d variables

- 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`**

rc.d script name is substituted to *daemon* in the variable name

daemon_class

- set to a login class of the same name as the rc.d script
- ~~netsnmpd_class=myclass~~

```
netsnmpd:\
    :openfiles-cur=512:\
    :tc=daemon:
```

rc.conf.local example

```
apmd_flags=-A

hotplugd_flags=

saned_flags=-s128

pkg_scripts=messagebus saned cupsd
```

Special cases

- meta rc.d script
  - `/etc/rc.d/samba start`
  - `/etc/rc.d/smdb start && /etc/rc.d/nmbd start`

Special cases

- multiple instances of the same daemon
  - ln -f /etc/rc.d/foobar /etc/rc.d/foobar2
  - pgrep(1) much match the correct one!

/etc/rc.d/rc.subr

- entry point
- where the whole framework is defined
- sourced by rc.d scripts
  - to get std functions and default vars
  - can be overridden by the script itself

rc_start()

```
${rcexec} "${daemon} ${daemon_flags} ${_bg}"

rcexec="su -l -c ${daemon_class} -s /bin/sh
        ${daemon_user} -c"

rc_bg=YES -> “&”

e.g.
su -l -c daemon -s /bin/sh root \
    -c "/usr/sbin/sshd -flags"
```

rc_stop()

```
pkill -xf "${pexp}"

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

rc_reload()

```
pkill -HUP -xf "${pexp}"
```

rc_check()

```
pgrep -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

Optional function: rc_post()

- invoked by rc_stop() after a daemon process has been killed
- cleanup
  - remove dangling lock files
  - putting the system back into a pristine state (e. g. cups)

Unsupported actions

- 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

rc_cmd()

- main function
- last command called by an rc.d script
- 1 of 5 arguments

rc_cmd() start

- check that the daemon is enabled
- check it is not already running
- run rc_pre()
- run rc_start()
- pexp in /var/run/rc.d/${daemon}
- wait up to ${daemon_timeout} seconds

rc_cmd() stop

- check that the daemon is running
- run rc_stop()
- wait up to ${daemon_timeout} seconds
- run rc_post()

rc_cmd() restart

- /etc/rc.d/daemon stop
- /etc/rc.d/daemon start

rc_cmd() reload

- check that the daemon is running
- run rc_reload()

rc_cmd() check

- rc_check()

rcctl(8)

- rc.conf.local "editor"
- configure & control daemons and services
- ala service(8) + chkconfig(8) + sysconfig
- alternative, not an $EDITOR replacement

rcctl - confusion achieved

```
multicast=YES
sshd=YES


multicast=
sshd_flags=


multicast_flags=NO
sshd_flags=NO
```

rcctl - coherence

- unified interface
- abstraction
- daemon versus service
- regular versus meta script
- rcctl support in Puppet, Ansible and Salt
  - puppet: 120 additions and 441 deletions

rcctl - sourcing

rcctl -> rc.subr -> rc.d script -> rc.conf+rc.conf.local -> rc.subr

- from sourced to parsed: _rc_parse_conf()
- stop injecting shell code in dangerous places

rcctl - usage

```
usage:  rcctl get|getdef|set service | daemon [variable [args]]
        rcctl [-df] action daemon ...
        rcctl disable|enable|order [daemon ...]
        rcctl ls lsarg
```

“rcctl ls faulty” is run daily(8)

Conclusion

- ! replacement for the traditional BSD init
- ! process control framework
- ! service supervisor
- compromise
  - may not be suitable for all possible uses

Conclusion

- boringly simple and robust
- preserved the original paradigm
- built on decades-old components
- consistent and unified interface with rcctl
- easy integration into other OSes

Thank you for listening

Questions ?

Thank you AsiaBSDCon

Antoine Jacoutot

<ajacoutot@openbsd.org>

The OpenBSD Project