1-800-RC(8)-HELP

Dial Into FreeBSD Service Scripts Mastery!

Mateusz Piotrowski <0mp@FreeBSD.org>

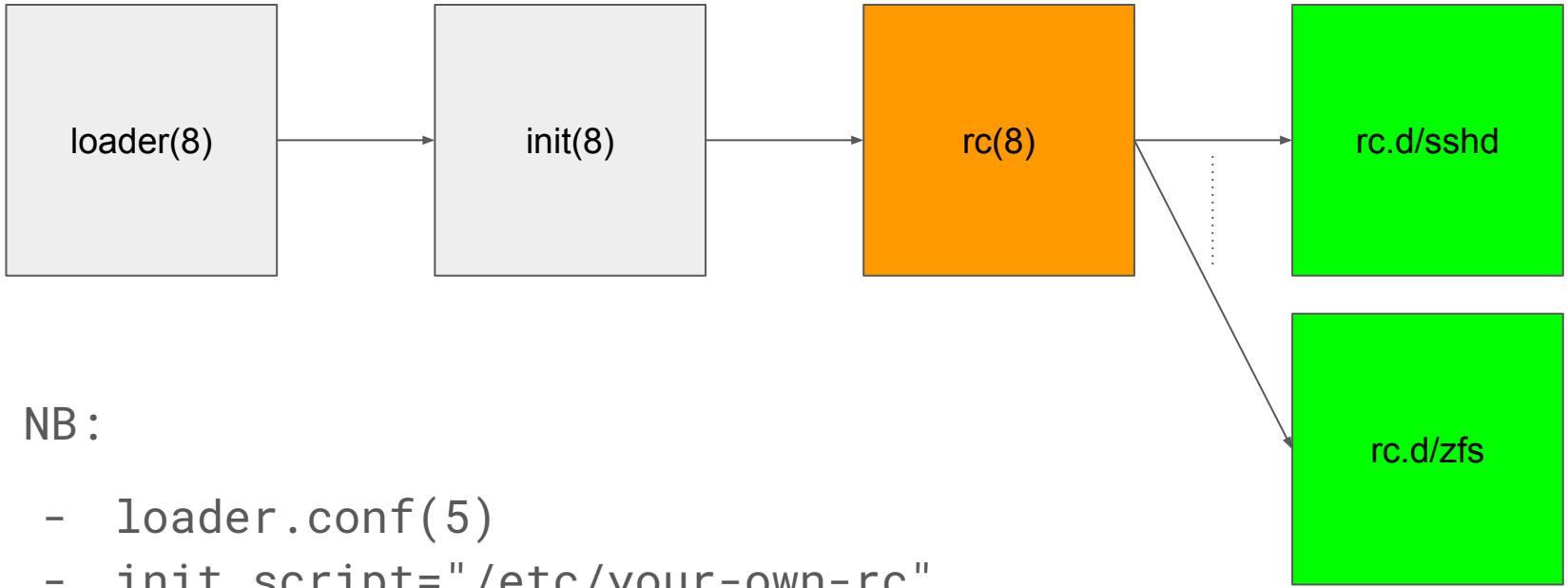
Getting Comfortable with Navigating rc(8) Ecosystem

Outline

1. Booting Process
2. Basic Usage
3. rc(8) Framework
4. Scripts Implementation
5. Configuration
6. Writing Scripts
7. Debugging Scripts
8. Optimizing Performance
9. Testing
10. Security News
11. Summary

Booting Process

Booting FreeBSD



NB:

- loader.conf(5)
- init_script="/etc/your-own-rc"

Basic Usage

How To Enable a Service (e.g., sshd)?

1. `echo sshd_enable=YES >> /etc/rc.conf`
2. `echo sshd_enable=YES >> /etc/rc.conf.d/sshd`
3. `/etc/rc.d/sshd enable`
4. `service sshd enable`
5. `sysrc sshd_enable=YES`

How To Restart a Service?

1. `/etc/rc.d/sshd restart`
2. `service sshd restart`

How To $ACTION a Service?

```
# service sshd status
sshd is running as pid 1141.

# service sshd poll
Waiting for PIDS: 1141

# service sshd stop
Stopping sshd.
Waiting for PIDS: 1141.

# service sshd extracommands
configtest keygen reload
```

rc(8) Framework

rc(8): Intro

- /etc/rc
  - Controls the booting process after init(8)
  - Runs all the services during boot
  - Handles special configuration setups like diskless booting
- rc framework
  - Common name for the programs helping with service management
  - Covers everything from boot to shutdown
  - Fairly small

/etc/rc: Implementation

```
# Handle: autobooting, diskless booting,
#         debugging, being jailed, firstboot.
# Then, run scripts:

…

files=`rcorder ${skip} … ${system_rc} …`
run_rc_scripts --break ${early_late_divider} … $files

…

*)   find_local_scripts_new ;;

…

files=`rcorder ${skip} … ${system_rc} ${local_rc} …`
run_rc_scripts … $files
```

rc.d: Service Scripts

```
$ ls /etc/rc.d
```

```
DAEMON          cleanvar        growfs_fstab    iscsid          mountd          power_profile   sendmail        virecover
FILESYSTEMS     cleartmp        gssd            jail            mountlate       powerd          serial          watchdogd
LOGIN           cron            hastd           kadmind         moused          ppp             sshd            wpa_supplicant
NETWORKING      ctld            hcsecd          kdc             msgs            pppoed          statd           ypbind
SERVERS         ddb             hostapd         keyserv         natd            pwcheck         static_arp      ypldap
accounting      defaultroute    hostid          kfd             netif           quota           static_ndp      yppasswdd
adjkerntz       devd            hostid_save     kld             netoptions      random          stf             ypserv
apm             devfs           hostname        kldxref         netwait         rarpd           swap            ypset
auditd          devmatch        inetd           kpasswdd        newsyslog       rctl            swaplate        ypupdated
auditdistd      dhclient        iovctl          ldconfig        nfscbd          resolv          syscons         ypxfrd
automount       dmesg           ip6addrctl      linux           nfsclient       rfcomm_pppd_server sysctl       zfs
automountd      dnctl           ipfilter        local           nfsd            root            sysctl_lastload zfsbe
autounmountd    dumpon          ipfs            local_unbound   nfsuserd        route6d         syslogd         zfsd
bgfsck          fsck            ipfw            localpkg        nisdomain       routed          sysvipc         zfskeys
blacklistd      ftp-proxy       ipfw_netflow    lockd           nscd            routing         tlsclntd        zpool
bluetooth       ftpd            ipmon           lpd             ntpd            rpcbind         tlsservd        zpoolreguid
bootparams      gbde            ipnat           mdconfig        ntpdate         rtadvd          tmp             zpoolupgrade
bridge          geli            ippool          mdconfig2       opensm          rtsold          ubthidhci       zvol
bsnmpd          geli2           ipropd_master   mixer           os-release      rwho            ugidfw
bthidd          ggated          ipropd_slave    motd            pf              savecore        utx
ccd             gptboot         ipsec           mountcritlocal  pflog           sdpd            var
cfumass         growfs          iscsictl        mountcritremote pfsync          securelevel     var_run
```

```
$ ls /usr/local/etc/rc.d
```

```
avahi-daemon    avahi-dnsconfd  dbus            git_daemon      samba_server
```

rcorder(8): Dependency Ordering

```
$ rcorder /etc/rc.d/* | grep -C 3 sshd

/etc/rc.d/LOGIN
/etc/rc.d/nfsd
/etc/rc.d/sendmail
/etc/rc.d/sshd
/etc/rc.d/ftpd
/etc/rc.d/inetd
/etc/rc.d/cron
```

rcorder(8): Placeholders

```
$ grep -A  2 PROVIDE rc.d/sshd

# PROVIDE: sshd
# REQUIRE: LOGIN FILESYSTEMS
# KEYWORD: shutdown

$ rcorder /etc/rc.d/* | grep -e LOGIN -e sshd

/etc/rc.d/LOGIN
/etc/rc.d/sshd

$ ls /etc/rc.d/[A-Z]*

DAEMON
FILESYSTEMS
LOGIN
NETWORKING
SERVERS
```

rcorder(8): BEFORE

```
$ grep -A 2 PROVIDE sshd

# PROVIDE: sshd
# REQUIRE: LOGIN FILESYSTEMS
# KEYWORD: shutdown
```

```
$ grep -A 2 PROVIDE sshd_preparer

# PROVIDE: sshd_preparer
# BEFORE: sshd
```

Other rc.* Scripts

```
rc.local

rc.firewall

rc.resume

rc.suspend

rc.shutdown

rc.shutdown.local

rc.final
```

Scripts Implementation

/etc/rc.d/sysctl: One-Off Pattern

```
#!/bin/sh
# PROVIDE: sysctl

. /etc/rc.subr

name="sysctl"
desc="Set sysctl variables …"
command="/sbin/sysctl"
stop_cmd=":"
start_cmd="sysctl_start"

…

sysctl_start() {
            …
            ${command} ${command_args} ${_f} …
            …
}

load_rc_config $name
…
run_rc_command "$1"
```

/etc/rc.d/inetd: Simple-Service Pattern

```
#!/bin/sh
# PROVIDE: inetd
# REQUIRE: DAEMON LOGIN FILESYSTEMS
# KEYWORD: shutdown

. /etc/rc.subr

name="inetd"
desc="Internet \"super-server\""
rcvar="inetd_enable"
command="/usr/sbin/${name}"
pidfile="/var/run/${name}.pid"
required_files="/etc/${name}.conf"
extra_commands="reload"

…

load_rc_config $name
run_rc_command "$1"
```

Configuration

rc.conf(5)

- /etc/defaults/rc.conf
- /etc/defaults/vendor.conf
- /etc/rc.conf
- /etc/rc.conf.local
- /etc/rc.conf.d/*${name}*
- /usr/local/etc/rc.conf.d/*${name}*

sysrc(8)

```
$ sysrc sshd_enable
sshd_enable: YES

# sysrc sshd_enable=YES
sshd_enable: YES -> YES

# sysrc cloned_interfaces+=gif0
cloned_interfaces:  -> gif0

# sysrc cloned_interfaces+='lo1 gif0'
cloned_interfaces: gif0 -> gif0 lo1

# sysrc -x cloned_interfaces
```

rc.conf(5) Is Just sh(1): lagg(4) Configuration

```
ifconfig_em0="up"

wlans_iwm0="wlan0"

ifconfig_wlan0="WPA powersave"

_mac="$(ifconfig em0 ether | awk '/ether/{print $2}')"

create_args_wlan0="wlanaddr ${_mac}"

cloned_interfaces="lagg0"

ifconfig_lagg0="up laggproto failover laggport em0 laggport wlan0 DHCP"
```

rc.conf(5) Is Just sh(1): Case Statements

```
case 0 in
0)
    ifconfig_em0="up"
    ;;
1)
    wlans_iwm0="wlan0"
    ifconfig_wlan0="WPA powersave"
    ;;
esac
```

Writing Scripts

rc.subr(8)

Functionalities targeted at developers:

- name
- rcvar
- command
- command_args
- sig_reload
- command_interpreter
- run_rc_command()
- run_rc_script()
- load_rc_config()
- force_depend()

Functionalities targeted at operators:

- ${name}_program
- ${name}_flags
- ${name}_oomprotect
- ${name}_nice

For example:

sshd_program="/usr/local/sbin/customsshd"

rc.subr(8): run_rc_command start | restart | reload

What happens during **run_rc_command()** of */etc/rc.d/foobar start*?

1. setup
2. prestart
3. start
4. poststart

Recent Additions: ${name}_offcmd & ${name}_setup

```
devd_offcmd=devd_off

…

devd_off() {
	# If devd is disabled, turn it off
	# in the kernel to avoid
	# unnecessary memory usage.
	if ! checkyesno ${rcvar}; then
		$SYSCTL hw.bus.devctl_queue=0
	fi
}
```

```
# Create the unbound configuration file
# and update resolv.conf to
# point to unbound.
#
local_unbound_setup() {
	…
}

#
# Before starting, check that the
# configuration file and root anchor
# exist.  If not, attempt to generate
them.
#
local_unbound_prestart() {
	…
}
```

openconnect: Multi-Instance Pattern (1/2)

```
# service openconnect start myvpn

# service openconnect start homevpn

# service openconnect start workvpn
```

openconnect: Multi-Instance Pattern (2/2)

```
# sysrc openconnect_myvpn_enable="YES"

# sysrc openconnect_myvpn_username="charlie.root"

# sysrc openconnect_myvpn_server="vpn.example.org"

# service openconnect setpassword myvpn
Password (openconnect_myvpn):

# sysrc openconnect_services+="myvpn"
```

rc.d/openconnect: Multi-Instance Pattern

```
_service="${2-default}"

name="openconnect_${_service}"
rcvar="openconnect_${_service}_enable"

load_rc_config "${name}"
…
pidfile="${_rundir}/${_service}.pid"

eval : "\${openconnect_${_service}_enable:=NO}"
…
eval _args=\$openconnect_${_service}_args
eval _server=\$openconnect_${_service}_server
…
procname="/usr/local/sbin/openconnect"
command="/usr/sbin/daemon"
command_args="-o /var/log/${name}.log -p ${pidfile} -- ${procname}"
…
command_args="${command_args} ${_username} ${_args} ${_server}"
sig_reload="SIGUSR2"
…
run_rc_command "$1"
```

rc.d/openconnect_services: Multi-Instance Pattern

```
# PROVIDE: openconnect_services
# REQUIRE: NETWORKING
# KEYWORD: shutdown

. /etc/rc.subr

name="openconnect_services"

load_rc_config "${name}"

for _service in ${openconnect_services}; do
    if ! /usr/local/etc/rc.d/openconnect "$1" "$_service"; then
        err 1 "Failed to complete command \"$1\" for service \"$_service\""
    fi
done
```

/usr/local/etc/rc.d/openconnect: Daemon(8) Pattern

```
…

pidfile="${_rundir}/${_service}.pid"

…

procname="/usr/local/sbin/openconnect"
command="/usr/sbin/daemon"
command_args="-o /var/log/${name}.log -p ${pidfile} -- ${procname}"

…

run_rc_command "$1"
```

Restart Scripts Automatically: daemon(8)

```
procname="/usr/local/bin/grafana"
command="/usr/sbin/daemon"
command_args="… -p ${pidfile} … \
              ${procname} …"
```

```
procname="/usr/local/bin/grafana"
command="/usr/sbin/daemon"
command_args="… -p ${pidfile} … \
              -R 5 \
              -P ${daemon_pidfile}\
              ${procname} …"
```

Debugging Scripts

Debugging: sh -x

```
$ sh -x /etc/rc.d/foobar …
+ . /etc/rc.subr
+ : 28812
+ export RC_PID
+ [ -n '' ]
+ _rc_subr_loaded=YES
+ SYSCTL=/sbin/sysctl
+ SYSCTL_N='/sbin/sysctl -n'
+ SYSCTL_W=/sbin/sysctl
+ PROTECT=/usr/bin/protect
+ ID=/usr/bin/id
```

Debugging: rc_debug

```
$ env rc_debug=1 /etc/rc.d/utx describe
/etc/rc.d/utx: DEBUG: Sourcing /etc/defaults/rc.conf
Manage the user accounting database
```

Debugging: debug.sh(8) (FreeBSD 15.0-CURRENT)

- Features include:
  - Enable set -x selectively
  - Drop into an interactive shell
- Supposedly, in use for already 30 years.
- Contributed to FreeBSD in February earlier this year

Optimizing Performance

Benchmarking service scripts with boottrace(4): Intro

```
# Enable boottrace(4) via loader.conf(5).
echo kern.boottrace.enabled=1 >> /boot/loader.conf


# Then reboot and let boottrace(4) collect the log.
# Access the collected logs with:
sysctl kern.boottrace.log
```

Benchmarking service scripts with boottrace(4): Log

```
CPU      msecs      delta process               event                                PID    CPUtime IBlks   OBlks
  0   44872811          0 kernel                sysinit 0x2100001                      0       0.00     0       0
  0   44872812          1 kernel                sysinit 0x2110000                      0       0.00     0       0
  0   44872812          0 kernel                sysinit 0x2140000                      0       0.00     0       0
…
  0   44872817          0 kernel                sysinit 0x2800000                      0       0.00     0       0
  0   44873820       1003 kernel                sysinit 0x2880000                      0       0.00     0       0
  0   44873820          0 kernel                sysinit 0x2888000                      0       0.00     0       0
…
  1   44875735          0 kernel                sysinit 0xffffffff                     0       0.00     0       0
  1   44875735          0 swapper               mi_startup done                        0       0.00     0       0
  0   44875750         15 init                  init(8) starting...                    1       0.00     0       0
  0   44875751          1 init                  /etc/rc starting...                    1       0.00     0       0
  0   44875831         80 boottrace             /etc/rc.d/rctl start                  26       0.00     0       0
  1   44875839          8 boottrace             /etc/rc.d/rctl done                   26       0.00     2       0
…
  0   44876446          0 boottrace             /etc/rc.d/netif start                390       0.00     0       0
  1   44881116       4670 boottrace             /etc/rc.d/netif done                 390       0.12    34       0
…
  0   44882866          1 boottrace             /etc/rc.d/securelevel start         1679       0.00     0       0
  0   44882872          6 boottrace             /etc/rc.d/securelevel done          1679       0.00     0       0
  1   44882879          7 init                  /etc/rc finished                       1       2.22   743      15
Total measured time: 10068 msecs

CPU      msecs      delta process               event                                PID    CPUtime  IBlks  OBlks
  1   44882880          0 init                  multi-user start                       1       2.22   743      15
  0   44918215      35335 kldload               hwpmc.ko: sysinit 0xd800000         1698       0.00     0      0
Total measured time: 35335 msecs
```

Testing

rc(8) Tests in The FreeBSD Test Suite

- In the src tree:
  - libexec/rc/tests
- Run tests on an installed system with:
  - kyua test -k /usr/tests/Kyuafile libexec/rc
- Great way to contribute to FreeBSD

Security News

Jailed services

```
# rc.conf(5):

${name}_svcj="YES"

${name}_svcj_options=net_basic

svcj_all_enable="YES"

# rc.subr(8):

$JAIL_CMD -c $_svcj_generic_params $_svcj_cmd_options \
      exec.start="${SERVICE} -E _rc_svcj=jailing ${name} ${_rc_prefix}start …" \
      exec.stop="${SERVICE} -E _rc_svcj=jailing ${name} ${_rc_prefix}stop …" \
      exec.consolelog="/var/log/svcj_${name}_console.log" \
      name=svcj-${name}
```

Summary

Today We Learnt

- rc(8) ecosystem is made of many replaceable parts
- rc(8) ecosystem is welcoming towards custom solutions
- Common service script patterns are:
  - One-off pattern
  - Simple service pattern
  - Multi-instance pattern
  - Daemon(8) pattern
- Recent development brought improvements in:
  - Performance
  - Debugging experience
  - Security
  - Customizability

1-800-THX-TTYL

Do you have any questions?