NAME

rc - command scripts for auto-reboot and daemon startup

SYNOPSIS

rc
rc.conf
rc.conf.local
rc.d/
rc.firewall
rc.local
rc.resume
rc.shutdown
rc.subr

DESCRIPTION

The **rc** utility is the command script which controls the automatic boot process after being called by init(8). The **rc.local** script contains commands which are pertinent only to a specific site. Typically, the */usr/local/etc/rc.d/* mechanism is used instead of **rc.local** these days but if you want to use **rc.local**, it is still supported. In this case, it should source */etc/rc.conf* and contain additional custom startup code for your system. The best way to handle **rc.local**, however, is to separate it out into **rc.d/** style scripts and place them under */usr/local/etc/rc.d/*. The **rc.conf** file contains the global system configuration information referenced by the startup scripts, while **rc.conf.local** contains the local system configuration. See rc.conf(5) for more information.

The **rc.d/** directories contain scripts which will be automatically executed at boot time and shutdown time.

The service(8) command provides a convenient interface to manage rc.d services.

The sysrc(8) command provides a scripting interface to modify system config files.

Operation of rc

1. If autobooting, set *autoboot*=yes and enable a flag (*rc_fast*=yes), which prevents the **rc.d/** scripts from performing the check for already running processes (thus speeding up the boot process). This *rc_fast*=yes speedup will not occur when **rc** is started up after exiting the single-user shell.

2. Determine whether the system is booting diskless, and if so run the */etc/rc.initdiskless* script.

3. Source */etc/rc.subr* to load various rc.subr(8) shell functions to use.

4. Load the configuration files (see below for reloading).

5. Determine if booting in a jail, and add "nojail" (no jails allowed) or "nojailvnet" (only allow vnet-enabled jails) to the list of KEYWORDS to skip in rcorder(8).

6. If the file *${firstboot_sentinel}* does not exist, add "firstboot" to the list of KEYWORDS to skip in rcorder(8).

7. Invoke rcorder(8) to order the files in */etc/rc.d/* that do not have a "nostart" KEYWORD (refer to rcorder(8)'s **-s** flag).

8. Call each script in turn using **run_rc_script**() (from rc.subr(8)), which sets *$1* to "start", and sources the script in a subshell. Stop processing when the script that is the value of the *$early_late_divider* has been run.

9. Check again to see if the file *${firstboot_sentinel}* exists (in case it is located on a newly mounted file system) and adjust the list of KEYWORDs to skip appropriately.

10. Re-run rcorder(8), this time including the scripts in the *$local_startup* directories. Ignore everything up to the *$early_late_divider*, then start executing the scripts as described above.

11. If the file *${firstboot_sentinel}* exists, delete it. If the file *${firstboot_sentinel}-reboot* also exists (because it was created by a script), then delete it and reboot.

Operation of rc.shutdown

1. Set *rc_shutdown* to the value of the first argument passed to **rc.shutdown** or to "unspecified" if no argument was passed.

2. Source */etc/rc.subr* to load various rc.subr(8) shell functions to use.

3. Load the configuration files.

4. Invoke rcorder(8) to order the files in */etc/rc.d/* and the *$local_startup* directories that have a "shutdown" KEYWORD (refer to rcorder(8)'s **-k** flag), reverse that order, and assign the result to a variable.

5. Call each script in turn using **run_rc_script**() (from rc.subr(8)), which sets *$1* to "faststop", and sources the script in a subshell.

Contents of rc.d/

rc.d/ is located in */etc/rc.d/*. The following file naming conventions are currently used in **rc.d/**:

ALLUPPERCASE Scripts that are "placeholders" to ensure that certain operations are performed before others. In order of startup, these are:

FILESYSTEMS Ensure that root and other critical file systems are mounted. This is the default *$early_late_divider*.

NETWORKING Ensure basic network services are running, including general network configuration.

SERVERS Ensure basic services exist for services that start early (such as *nisdomain*), because they are required by *DAEMON* below.

DAEMON Check-point before all general purpose daemons such as *lpd* and *ntpd*.

LOGIN Check-point before user login services (*inetd* and *sshd*), as well as services which might run commands as users (*cron* and *sendmail*).

bar Scripts that are sourced in a subshell. The boot does not stop if such a script terminates with a non-zero status, but a script can stop the boot if necessary by invoking the **stop_boot**() function (from rc.subr(8)).

Each script should contain rcorder(8) keywords, especially an appropriate "PROVIDE" entry, and if necessary "REQUIRE" and "BEFORE" keywords.

Each script is expected to support at least the following arguments, which are automatically supported if it uses the **run_rc_command**() function:

start Start the service. This should check that the service is to be started as specified by rc.conf(5). Also checks if the service is already running and refuses to start if it is. This latter check is not performed by standard FreeBSD scripts if the system is starting directly to multi-user mode, to speed up the boot process. If **forcestart** is given, ignore the rc.conf(5) check and start anyway.

stop If the service is to be started as specified by rc.conf(5), stop the service. This should check

that the service is running and complain if it is not. If **forcestop** is given, ignore the rc.conf(5) check and attempt to stop.

restart Perform a **stop** then a **start**.

status If the script starts a process (rather than performing a one-off operation), show the status of the process. Otherwise it is not necessary to support this argument. Defaults to displaying the process ID of the program (if running).

enable
Enable the service in rc.conf(5).

disable
Disable the service in rc.conf(5).

delete Remove the service from rc.conf(5). If 'service_delete_empty' is set to "YES", */etc/rc.conf.d/$servicename* will be deleted if empty after modification.

describe
Print a short description of what the script does.

extracommands
Print the script's non-standard commands.

poll If the script starts a process (rather than performing a one-off operation), wait for the command to exit. Otherwise it is not necessary to support this argument.

enabled
Return 0 if the service is enabled and 1 if it is not. This command does not print anything.

rcvar Display which rc.conf(5) variables are used to control the startup of the service (if any).

If a script must implement additional commands it can list them in the *extra_commands* variable, and define their actions in a variable constructed from the command name (see the *EXAMPLES* section).

The configuration files are normally read only once at the start of a boot sequence; if a script needs to **enable** or **disable** any other script that would run later in the sequence, it must send a SIGALRM to the rc process (identified by $RC_PID) to have it re-read the files.

The following key points apply to old-style scripts in */usr/local/etc/rc.d/*:

- Scripts are only executed if their basename(1) matches the shell globbing pattern **.sh*, and they are executable. Any other files or directories present within the directory are silently ignored.

- When a script is executed at boot time, it is passed the string "start" as its first and only argument. At shutdown time, it is passed the string "stop" as its first and only argument. All **rc.d/** scripts are expected to handle these arguments appropriately. If no action needs to be taken at a given time (either boot time or shutdown time), the script should exit successfully and without producing an error message.

- The scripts within each directory are executed in lexicographical order. If a specific order is required, numbers may be used as a prefix to the existing filenames, so for example *100.foo* would be executed before *200.bar*; without the numeric prefixes the opposite would be true.

- The output from each script is traditionally a space character, followed by the name of the software package being started or shut down, *without* a trailing newline character.

SCRIPTS OF INTEREST

When an automatic reboot is in progress, **rc** is invoked with the argument **autoboot**. One of the scripts run from */etc/rc.d/* is */etc/rc.d/fsck*. This script runs fsck(8) with option **-p** and **-F** to "preen" all the disks of minor inconsistencies resulting from the last system shutdown. If this fails, then checks/repairs of serious inconsistencies caused by hardware or software failure will be performed in the background at the end of the booting process. If **autoboot** is not set, when going from single-user to multi-user mode for example, the script does not do anything.

The */etc/rc.d/local* script can execute scripts from multiple **rc.d/** directories. The default location includes */usr/local/etc/rc.d/*, but these may be overridden with the *local_startup* rc.conf(5) variable.

The */etc/rc.d/serial* script is used to set any special configurations for serial devices.

The **rc.firewall** script is used to configure rules for the kernel based firewall service. It has several possible options:

open will allow anyone in
client will try to protect just this machine
simple will try to protect a whole network
closed totally disables IP services except via *lo0* interface
UNKNOWN
disables the loading of firewall rules
filename will load the rules in the given filename (full path required).

Most daemons, including network related daemons, have their own script in */etc/rc.d/*, which can be used to start, stop, and check the status of the service.

Any architecture specific scripts, such as */etc/rc.d/apm* for example, specifically check that they are on that architecture before starting the daemon.

Following tradition, all startup files reside in */etc*.

FILES

/etc/rc
/etc/rc.conf
/etc/rc.conf.local
/etc/rc.d/
/etc/rc.firewall
/etc/rc.local
/etc/rc.shutdown
/etc/rc.subr
/var/run/dmesg.boot dmesg(8) results soon after the **rc** process begins. Useful when dmesg(8) buffer in the kernel no longer has this information.

EXAMPLES

The following is a minimal **rc.d/** style script. Most scripts require little more than the following.

```
#!/bin/sh
#

# PROVIDE: foo
# REQUIRE: bar_service_required_to_precede_foo

. /etc/rc.subr

name="foo"
rcvar=foo_enable
command="/usr/local/bin/foo"

load_rc_config $name
run_rc_command "$1"
```

Certain scripts may want to provide enhanced functionality. The user may access this functionality through additional commands. The script may list and define as many commands at it needs.

```
#!/bin/sh
#

# PROVIDE: foo
# REQUIRE: bar_service_required_to_precede_foo
# BEFORE:  baz_service_requiring_foo_to_precede_it

. /etc/rc.subr

name="foo"
rcvar=foo_enable
command="/usr/local/bin/foo"
extra_commands="nop hello"
hello_cmd="echo Hello World."
nop_cmd="do_nop"

do_nop()
{
        echo "I do nothing."
}

load_rc_config $name
run_rc_command "$1"
```

As all processes are killed by init(8) at shutdown, the explicit kill(1) is unnecessary, but is often included.

SEE ALSO

kill(1), rc.conf(5), init(8), rc.resume(8), rc.subr(8), rcorder(8), reboot(8), savecore(8), service(8), sysrc(8)

Practical rc.d scripting in BSD, https://docs.freebsd.org/en/articles/rc-scripting/.

HISTORY

The **rc** utility appeared in 4.0BSD.