OpenRC

OpenRC

OpenRC is a dependency-based init system that initially maintained compatibility with sysvinit and, beginning with OpenRC 0.25, replaced `/sbin/init` with its own program. OpenRC was written by Gentoo developers and it is designed to be used by other distributions and BSD systems.

Contents

Installation

Install the `openrc` package.

Installation of services

OpenRC service packages are named *`package_name`*`-openrc` and, when installed, are normally available in `/etc/init.d`.

Features

OpenRC provides features such as hardware initiated initscripts or cgroups support.

Configuration

`/etc/rc.conf` is the global OpenRC configuration file.

Network management

OpenRC can be used with one of several network managers or even with none, see `/etc/conf.d/net` for examples of static or dynamic network configuration. The **netifrc** package provides a collection of modules for configuring and managing network interfaces via individual, per-interface scripts located in the /etc/init.d/ directory. The package supplies only `/etc/init.d/net.lo` (for the loopback interface), but doesn't hard-code operations on namely **lo** and the same piece can control other interfaces (as `net.`*`interface_name`* `-> net.lo`) as a cheap alternative to deploying separate scripts. **netifrc** is not mandatory for OpenRC – it can be left unused in favour of another network manager, or just be missing.

Dependency behavior

Changing the default dependencies of init scripts might be needed to fit more complex setups. See `/etc/rc.conf` for how to change the default behavior; notice the **rc_depend_strict** option. In addition, the following networking examples show how flexible OpenRC can be.

Selecting a specific runlevel at boot

OpenRC reads the kernel command-line used at boot time, and will start the runlevel specified by the **softlevel** parameter if provided, instead of **default**. For instance, you can choose whether to boot into the **default**, **nonetwork** or **single-user** runlevels with the following example grub.conf configuration:

```
 /boot/grub/grub.conf
 title=Regular start-up
 kernel (hd0,0)/boot/vmlinuz-linux root=/dev/sda3
 title=Start without networking
 kernel (hd0,0)/boot/vmlinuz-linux root=/dev/sda3
softlevel=nonetwork
 title=Single-user mode
 kernel (hd0,0)/boot/vmlinuz-linux root=/dev/sda3 softlevel=single
```

Usage

Runlevels

OpenRC uses runlevels in very much the same way as sysvinit (or BSD init). At any given time the system is in one of the defined runlevels. There are three internal runlevels and four user defined runlevels.

- Internal runlevels, the names are self-explanatory:
  - `sysinit`
  - `shutdown`
  - `reboot`
- User Runlevels:
  - `boot` : Starts all system-necessary services for other runlevels
  - `default` : Used for day-to-day-operations
  - `nonetwork` : Used when no network connectivity is required
  - `single` : Single-user mode

The system can switch to a runlevel with the **openrc <runlevel>** command, e.g.:

```
openrc nonetwork
```

Control

OpenRC can be controlled and configured using the **openrc**, **rc-service**, **rc-update** and **rc-status** commands.

Adding / removing services to / from runlevels

```
# rc-update add service runlevel
# rc-update del service runlevel
```

Start / restart / stop service

```
# rc-service service start
# rc-service service restart
# rc-service service stop
```

Listing

Use **rc-update show -v** to display all available init scripts and their current runlevel (if they have been added to one):

```
# rc-update show -v
```

Running **rc-update** or **rc-update show** will display only the init scripts that have been added to a runlevel.

Named runlevels

OpenRC runlevels are directories living in `/etc/runlevels`. To create additional runlevels, it is enough to issue:

```
# install -d /etc/runlevels/$runlevel
```

Stacked runlevels

It is possible to manage variants using **rc-update -s**. An example for using stacked runlevels on laptop to group networking services based on location is at OpenRC/StackedRunlevel.

Hotplug

OpenRC monitors and can be triggered by external events, such as new hardware from udev. See OpenRC/Event Driven for details.

Automatic respawning of crashed services

OpenRC can return the state of services to the runlevel's setting state, provide stateful init scripts and automatic respawning. If one issues **openrc** as root (for the default runlevel), crashed services will start and manually launched services will stop. To prevent the latter you can run **openrc -n** (--not-stop) By default openrc will attempt to just start crashed services, not restart. This is controlled by the **rc_crashed_stop** (default NO) and **rc_crashed_start** (default YES) options in `/etc/rc.conf`.

CGroups support

OpenRC has extended cgroups support. See OpenRC/CGroups for details.

rc.local

OpenRC doesn't execute `/etc/rc.local` by default; instead, it executes scripts from `/etc/local.d` ending in .start when local service starts and .stop when it stops. To mimic the good old `rc.local` behaviour, create `/etc/local.d/rc.local.start` with the following and make it executable with **chmod +x**:

```
# /etc/local.d/rc.local.start
[ -e /etc/rc.local ] && /etc/rc.local
```

Make sure that the **local** service is enabled. The scripts in `/etc/local.d` are executed in lexical order and the example above assumes `/etc/rc.local` has got execute permissions.

Examples

Multiple network interfaces

The SSH service must come up with the internal network, for instance eth0 and never wlan0. Override the **net** dependency from `/etc/init.d/sshd`, and refine it to depend on **net.eth0**:

```
/etc/conf.d/sshd
rc_need="!net net.eth0"
```

Multiple network interfaces in multiple runlevels

The SSH service must start with **eth0** (not **wlan0**) in the "default" runlevel, but in the "office" runlevel it must start with **wlan0** (not **eth0**).

```
/etc/rc.conf
#rc_depend_strict="YES"
```

Make additional symlinks to sshd with the network interface names:

```
# ln -s sshd /etc/init.d/sshd.eth0
# ln -s sshd /etc/init.d/sshd.wlan0
```

Settings are read from `/etc/conf.d/sshd.eth0` and `/etc/conf.d/sshd.wlan0` now:

```
# cp /etc/conf.d/sshd /etc/conf.d/sshd.eth0
# cp /etc/conf.d/sshd /etc/conf.d/sshd.wlan0
```

Add the dependencies:

```
# echo 'rc_need="!net net.eth0"' >> /etc/conf.d/sshd.eth0
# echo 'rc_need="!net net.wlan0"' >> /etc/conf.d/sshd.wlan0
```

In this example net.eth0 and net.wlan0 read their settings from `/etc/conf.d/net`, or `/etc/conf.d/net.office`, depending on the active runlevel. Add all runscripts to the different runlevels:

```
# rc-update add sshd.eth0 default
# rc-update add sshd.wlan0 office
# rc-update add net.eth0 default office
# rc-update add net.wlan0 default office
```

To switch between the "default" runlevel and the "office" runlevel without rebooting the computer, change to the "nonetwork" runlevel in between. The network interfaces will be stopped this way, and re-read their runlevel specific configuration. This works best when "nonetwork" is a stacked runlevel in both the "default" and "office" runlevels, and the display manager and other non-network services are added to the "nonetwork" runlevel only.
default runlevel <---> nonetwork runlevel <---> office runlevel

```
# rc nonetwork && rc office
# rc nonetwork && rc default
```

User Services

Generic user setup

The init scripts are loaded from `/etc/user/init.d`, and `${XDG_CONFIG_HOME}/rc/init.d`.

Runlevels are kept in `${XDG_CONFIG_HOME}/rc/runlevels`. `~/.config` is the default value should `XDG_CONFIG_HOME` be unset.

Managing services

Working with OpenRC user services is similar to system services, with the exception that commands **must not be run as root**, and they take a `--user` option (or `-U` for short).

Starting:
`$ rc-service <service-name> start --user`
Status check:
`$ rc-service <service-name> status --user`
Stopping:
`$ rc-service <service-name> stop --user`
Set up autostart:
`$ rc-update --user add <service-name> default`
Status of all user services:
`$ rc-status --user`

Creating a service

In order to set up a custom service, create a file named *service-name* in `${XDG_CONFIG_HOME}/rc/init.d` (create this folder if it doesn't exist), and make it executable:

```
$ mkdir -p ~/.config/rc/init.d
$ nano ~/.config/rc/init.d/service-name
$ chmod +x ~/.config/rc/init.d/service-name
```

You can use most features available in OpenRC system services, including custom loggers, dependencies, `supervisor="supervise-daemon"`, and others.
For an example, take a look at the official Artix scripts for wireplumber, pipewire or user-session dbus.

Configuration

Configurations in ~/.config/rc/conf.d should overwrite /etc/user/conf.d, and similarly options set in ~/.config/rc/rc.conf override /etc/rc.conf.

Manual setup needed for Pipewire/D-Bus

Until OpenRC PR #782 is merged, you must set `DBUS_SESSION_BUS_ADDRESS` manually.

If you launch your GUI through the TTY, this means you need to run

```
env "DBUS_SESSION_BUS_ADDRESS=${XDG_RUNTIME_DIR}/bus" startx
```

instead of just `startx`/the command to launch your Wayland compositor.

For GUI DMs, you'll need to create a custom entry. Instructions may vary.

Documentation

https://github.com/OpenRC/openrc/blob/master/user-guide.md#use

---