OpenRC init script in 15 minutes

Surprisingly easy and straightforward

Onkobu Tanaake

2020-04-16T10:00:00

At home and at work I often use VPN connections and need some setup scripts. I also need a bit of separation of concerns. Therefore it is not that comfortable for me to run connections as root. So I decided to split this in half, setup the tunnel device as root for a dedicated user. This user then works purely in user space. Tired of searching through bash history I looked up how to write a init script in Gentoo.

It is surprisingly easy: . This automatically inserts a skeleton into the editor – what a surprise. The only hard part was how to operate error codes and info messages. But first things first, the skeleton:

```
#!/sbin/openrc-run
# Copyright 2020 Gentoo Authors
# Distributed under the terms of the GNU General Public License v2

name="sample daemon"
description=""
command=/usr/bin/sample
command_args="${sample_args}"

depend() {

}
```

With Gentoo's wiki [https://wiki.gentoo.org/wiki/Handbook:AMD64/Working/Initscripts#Layout] is quite helpful filling the white space with something meaningful. So I came up with the following:

```
#!/sbin/openrc-run
# Copyright 2020 Gentoo Authors
# Distributed under the terms of the GNU General Public License v2

tun="tun0"
user="onkobu"

name="${user} tun daemon"
description="Creates ${tun} device for user ${user}"
command="ip tuntap add ${tun} mode tun user ${user}"
command_args="${tun_onkobu_args}"
```

```
depend() {
 need dnsmasq
}

start() {
 ebegin "Creating ${tun} for ${user}"
 ${command}
 eend $?
}

status() {
 ebegin "Checking ${tun}"
 ip link show ${tun} >/dev/null 2>&1
 eend $?
}

stop() {
 ebegin "Deleting ${tun}"
 ip link delete ${tun} >/dev/null 2>&1
 eend $?
}
```

A start()- and stop()-function got me going, setting up a tunnel device and deleting it. This ships with default restart()-function, calling stop() first followed by start(). This I already knew but I wasn't certain about state management. So I tried the script and was done: state management is included, stop() without preceeding start gives the corrent error.

One final piece was missing: configuring MTU (maximum transmission unit). This is per-user or sometimes per-device. And again configuration is already included. Each script automatically sources a file /etc/conf.d/<script-name> So I simply added an assignment to a config file with a bit of evaluation during startup:

```
start() {
 ebegin "Creating ${tun} for ${user}"
 ${command}
 if [ ! -z "${MTU}" ]; then
  # from /lib64/rc/bin
  einfo "With MTU ${MTU}"
  ip link set dev ${tun} mtu ${MTU}
 fi
 eend $?
}
```

Most of the time I spent searching for . This is not mentioned on the wiki page and there aren't many examples of the different commands. As regular user you're also not lucky, yields an error. But as root I found the correct place to search: /lib64/rc/bin. And so I found einfo and eindent. The fit perfectly and the MTU-message appears.

There wasn't anything similar easy in the last 3 months. And I was surprised how straightforward I came from the skeleton to the final script.