Understanding System V startup

Authored by: Brian E. Brzezicki

PALADIN
GROUP LLC

init

- Traditionally Unix systems have used the ATT System V initialization mechanism
- In System V init, there is a concept called a *runlevel*. The *runlevel* defines what services should start (or stop) upon entering or leaving the run level.

runlevels

- Different versions of Unix define the *runlevels* differently. In Red Hat Linux the *runlevels* are defined as below.

  runlevel 0 - system shutdown

  runlevel 1 - single user mode

  runlevel 2 – multiuser mode without networking

  runlevel 3 – multiuser mode with networking

  runlevel 4 – *unused*

  runlevel 5 – multiuser mode with networking and graphics (X11)

  runlevel 6 – system reboot

Services and init scripts

- Each service or sub system has a shell script that is responsible to start and stop the service. These scripts live in a directory called /etc/init.d

```
[root@paladin ~]# ls /etc/rc.d/init.d
abrt-ccpp      haldaemon          netfs          rpcsvcgssd
abrtd          halt               network        rsyslog
abrt-oops      htcacheclean       NetworkManager sandbox
acpid          httpd              nfs            saslauthd
atd            ip6tables          nfslock        single
auditd         iptables           ntpd           smartd
...
```

Running the scripts

- Each of these script can be run directly provided with an argument to **start, stop,** or **restart** the service.

```
[root@paladin ~]# /etc/rc.d/init.d/sshd stop
Stopping sshd:                                             [  OK  ]
[root@paladin ~]# /etc/rc.d/init.d/sshd start
Starting sshd:                                             [  OK  ]
```

- Most services support other actions such as **reload** or **status**

/etc/rc.d

- The directory /etc/rc.d contains a directory for each different *runlevel* on the system. Each directory will files or links to define what should start up or shutdown when entering that *runlevel.*

```
[root@paladin ~]# ls /etc/rc.d
init.d  rc0.d  rc2.d  rc4.d  rc6.d      rc.sysinit
rc      rc1.d  rc3.d  rc5.d  rc.local
```

/etc/rc.d/rc*X*.d

- The directory /etc/rc.d/rc*X*.d contains all of the startup scripts for each service that should be started or stopped at this runlevel.
- These scripts are usually simply symbolic links * back to the scripts in /etc/rc.d/init.d. Since there are multiple *runlevels* and services, this ensures the scripts only have to be located in one place.

```
[root@paladin ~]# ls -l /etc/rc.d/rc5.d
total 0
lrwxrwxrwx. 1 root root 20 Aug 26 16:32 K01certmonger -> ../init.d/certmonger
lrwxrwxrwx. 1 root root 23 Aug 26 16:34 K01matahari-host -> ../init.d/matahari-h
ost
```

* Note: Some versions of Unix use hard links rather than symbolic links

script naming

- Each link (or file) in /etc/rc*X*.d is named based on the following format

S##*service_name*

or

K##*service_name*

Example:

S55sshd

Breaking down the script names

- The first 3 characters in the script name defines much of the behavior of the service at this *runlevel*.
- The first character in the filename determines whether the service should be **started** or **stopped** upon entering the *runlevel.*
  - **K** = stop the service
  - **S** = start the service
- The next two characters are numbers and determine the order (lowest to highest) that the service will be started or stopped.
- **init** will first **stop** all services starting with K, in the defined order, before **starting** all services starting with S.
- If one service needs to be started (or stopped) before another, this is where the number is useful, to determine dependence

Example:
S55sshd will start AFTER S10network

/etc/rc.d/rc

- /etc/rc.d/rc is the script that actually runs the scripts. It is started by **init** and given the desired *runlevel* to enter as a command line argument. For each script in the /etc/rc.d/rc*X*.d directory it will call

  script **start** or *script* **stop**

  based on the filename.

The startup loop in /etc/rc.d/rc

```
# Now run the START scripts.
for i in /etc/rc$runlevel.d/S* ; do

        # Check if the subsystem is already up.
        subsys=${i#/etc/rc$runlevel.d/S??}
        [ -f /var/lock/subsys/$subsys ] && continue
        [ -f /var/lock/subsys/$subsys.init ] && continue
        check_runlevel "$i" || continue

        # If we're in confirmation mode, get user confirmation
        if [ "$do_confirm" = "yes" ]; then
                confirm $subsys
                rc=$?
                if [ "$rc" = "1" ]; then
                        continue
                elif [ "$rc" = "2" ]; then
                        do_confirm="no"
                fi
        fi

        update_boot_stage "$subsys"
        # Bring the subsystem up.
        [ -n "$UPSTART" ] && initctl emit --quiet starting JOB=$subsys
        if [ "$subsys" = "halt" -o "$subsys" = "reboot" ]; then
                export LC_ALL=C
                exec $i start
        fi
        $i start
```

Starting up your own processes

- Since the scripts in /etc/rc.d/init.d are simply shell scripts, you can create you own to startup your own programs at boot.
- Just make sure to create a startup script in /etc/rc.d/init.d that response to a **start** and **stop** argument.
- Then make the appropriate links in /etc/rc.d/rc*X*.d pointing back to your script in /etc/init.d

/etc/inittab

- The main configuration file for the **init** program itself is /etc/inittab. Processes can be configured to start directly in /etc/inittab. In fact the script that starts all the System V scripts (/etc/rc.d/rc) itself is configured in /etc/inittab.

```
l1:0:wait:/etc/rc.d/rc 0
l1:1:wait:/etc/rc.d/rc 1
l1:2:wait:/etc/rc.d/rc 2
l1:3:wait:/etc/rc.d/rc 3
l1:4:wait:/etc/rc.d/rc 4
l1:5:wait:/etc/rc.d/rc 5
l1:6:wait:/etc/rc.d/rc 6
```

/etc/inittab format

- The configuration file /etc/inittab defines processes that **init** should start and manage. The format of this text file is 1 line per process with multiple fields separated by a colon (:). The fields are defined below

Unique ID	Runlevel	Action	Process
1-4 characters which uniquely identify this line item.	The run levels where the process should be started	How to manage the process	The path to the process including any command line arguments

Moving forward

- Though **init** has been used for decades, it shows it's age. There have been different attempts to replace **init**. (**upstart**, **systemd** etc) Even distributions that use these new methods often follow the System V structure for backwards compatibility.

Knowledge is cool…