

LPIC-1 101-500 – Lesson 11

101.3 Change runlevels / boot targets and shutdown or reboot system



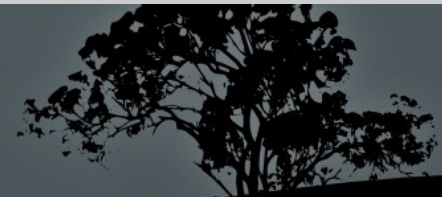
System V Init Runlevels

- The term **runlevel** refers to different modes of operation of UNIX and Linux systems that use System V init
- In the recent past most Linux distros used System V init. Some are still using it (Slackware, Gentoo)
- Recent versions of popular Linux distros have replaced System V init:
 - **systemd**: Fedora 15, CentOS 6, Debian 8, Ubuntu 16.04 and newer
 - **upstart**: Ubuntu 9.10 until 14.04



Typical Runlevels in Linux (Linux Standard Base – Red Hat)

ID	Name	Description
0	Halt	Power System Off
1,s,S	Single-User Mode	Recovery and admin mode. Used to repair corrupted filesystems, recover lost or corrupted files, recover root password, backup system etc. Network is inactive
2	Multi-User Mode	This mode supports multiple users, but networking and network interface cards are disabled
3	Multi-User Mode with Networking	Normal multiuser operation with networking enabled but no GUI
4	Unused/User Defined	For special cases
5	Multiuser with Networking and GUI	Normal multiuser operation with networking and GUI
6	Reboot	System Reboot



Runlevels in Legacy Debian/Ubuntu and Derivatives

ID	Name	Description
0	Halt	Power System Off
1,s,S	Single-User Mode	Recovery and admin mode. Used to repair corrupted filesystems recover lost or corrupted files, recover root password, backup system etc. Network is inactive
2-5	Multi-user with networking	Normal operation with GUI, if available. The default runlevel is 2
6	Reboot	System Reboot



Single-User Mode

- To enter into single user mode, you should pass one of these kernel parameters: **1**, **s**, **S** or **single**
 - For **GRUB legacy** press: Tab → **e** → choose **kernel** ... → **e** → add the parameter at the end e.g.: **s** → Enter → **b**
 - For **GRUB2** press Tab → **e** → choose **linux** (or **linux16**) → add the parameter at the end e.g.: **s** → **Ctrl-x**
- To enter into single user mode while the system is active:
init 1 # = init s, = init S

Note: on most systems the GRUB menu already provides a menuentry for Single User (or “recovery) mode

The */etc/inittab* file

- The ***/etc/inittab*** is very important on **sysvinit** systems, because it tells **init** which processes to run on startup, define the runlevels and monitoring of critical applications
- The format of the file is:

<id>:<runlevels>:<action>:<process>
- In modern distributions using **systemd** or **upstart** it is not used



The */etc/inittab* file

- # Set default runlevel. 2 for Debian, 3 for RedHat without GUI, 5 for RedHat with GUI

id:2:initdefault:

- # Action to be taken on pressing CTRL-ALT-DEL

ca:12345:ctrlaltdel:/sbin/shutdown -t1 -a -r now

- # What to do in single-user mode.

~~:S:wait:/sbin/sulogin



The */etc/inittab* file

- # Set the runlevels

```
si::sysinit:/etc/init.d/rcS
```

```
# The following lines define the runlevels (Debian, etc)
```

```
l0:0:wait:/etc/init.d/rc 0
```

```
l1:1:wait:/etc/init.d/rc 1
```

```
l2:2:wait:/etc/init.d/rc 2
```

```
l3:3:wait:/etc/init.d/rc 3
```

```
l4:4:wait:/etc/init.d/rc 4
```

```
l5:5:wait:/etc/init.d/rc 5
```

```
l6:6:wait:/etc/init.d/rc 6
```

- The `rc` script is responsible for calling the different runlevel scripts
- In RedHat systems `rc` is under `/etc/rc.d/rc`



The */etc/inittab* file

- # Start Terminals tty1 to tty6

1:2345:respawn:/sbin/getty 38400 tty1

2:23:respawn:/sbin/getty 38400 tty2

3:23:respawn:/sbin/getty 38400 tty3

4:23:respawn:/sbin/getty 38400 tty4

5:23:respawn:/sbin/getty 38400 tty5

6:23:respawn:/sbin/getty 38400 tty6



Important Files for System V init

- **/etc/rc.sysinit** (RedHat) or **/etc/init.d/rcS** (Debian): Prepare the system for startup. Executes before any other services
- **/etc/rc**, **/etc/rc.d/rc** (RedHat), **/etc/init.d/rc** (Debian): Used for changing runlevels
- **/etc/rc.local**: used for admin defined processes. The last script executed.



Important Directories for System V init

- `/etc/rc.d/init.d` (RedHat), `/etc/init.d` (Debian): Here we find the **stop** and **start** scripts of the various services
- `/etc/rc[0-6].d`: here we find various symlinks, back to `/etc/init.d` scripts, which defines the stopped and started services at each **runlevel**



Managing services/daemons in System V init

- All the management scripts are under `/etc/init.d`:

```
# ls -la /etc/init.d
```

- `# /etc/init.d/ssh stop` # stop the sshd daemon
- `# /etc/init.d/ssh start` # start the sshd daemon
- `# /etc/init.d/ssh restart` # restart the sshd daemon
- `# /etc/init.d/ssh reload` # reload configuration files for sshd (SIGHUP)
- `# /etc/init.d/ssh status` # status of the sshd daemon (active, inactive)

Managing services/daemons in System V init

- `# service <daemon> (start | stop | restart | reload | status)` # works in RedHat as well as recent versions of Debian/Ubuntu
- `# invoke-rc.d <daemon> (start | stop | restart | reload | status)` # for Debian/Ubuntu and derivatives



The directories /etc/rc[0-6].d

- `$ ls -la /etc/rc[0-6].d` # contain all the symbolic links to /etc/init.d script that defines what starts and what stops at each runlevel, e.g.:

/etc/rc1.d:

```
lrwxrwxrwx  1 root root    17 2011-09-03 15:26 K09apache2  
-> ../init.d/apache2
```

```
lrwxrwxrwx  1 root root   20 2011-08-27 08:21  
K15pulseaudio -> ../init.d/pulseaudio
```

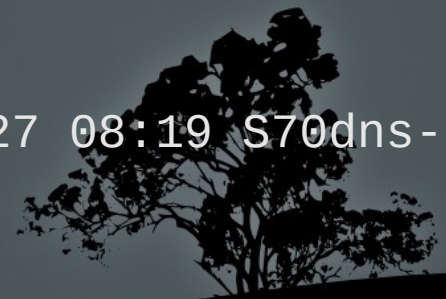
```
lrwxrwxrwx  1 root root   22 2011-08-27 08:19 K20acpi-  
support -> ../init.d/acpi-support
```

...

```
-rw-r--r--  1 root root   369 2009-09-07 21:58 README
```

```
lrwxrwxrwx  1 root root   19 2011-08-27 07:52  
S30killprocs -> ../init.d/killprocs
```

```
lrwxrwxrwx  1 root root   19 2011-08-27 08:19 S70dns-  
clean -> ../init.d/dns-clean
```



The directories /etc/rc[0-6].d

- `lrwxrwxrwx 1 root root 16 2011-08-27 07:52 S90single -> ../init.d/single`
- **S** is equivalent to /etc/init.d/single start
- **K** is equivalent to /etc/init.d/single stop
- For enabling or disabling a service we simple rename the symbolic link from **K** to **S** or from **S** to **K** respectively
- The **90** value sets the execution priority of the scripts. A smaller value represents a higher priority. The scripts in sysvinit are executed sequentially.

The commands `init` and `telinit`

- `# init 0 # power system off (runlevel 0)`
- `# init 6 # reboot system (runlevel 6)`
- `# init 1 # = init s, init S, enter single user mode (runlevel 1)`
- `# init 3 # enter runlevel 3`
- `# init 5 # enter runlevel 5`


*Note: on sysvinit the `/sbin/telinit` command is usually a symbolic link to `/sbin/init` and behaves in the same way. In **systemd** systems **`init`** points to **`systemd`** and **`telinit`** to **`systemctl`***

Show current runlevel with `runlevel`

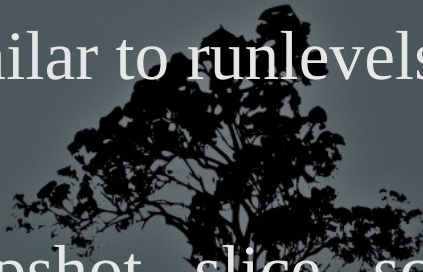
- `$ runlevel`
- `S 2 # previous runlevel: Single
current runlevel: 2`
- `2 3 # previous runlevel: 2
current runlevel: 3`
- `N 3 # previous runlevel: None!
current runlevel: 3`



The *systemd* init system

- **systemd** is a more powerful replacement for **sysvinit**
 - It provides concurrent startup of services
 - Services can be dependent on the status of other services
 - Services can be monitored and supervised
 - Separates the system resources into **units**
 - Replaces scripts with unit configuration files.
 - It is backward compatible with **sysvinit**
 - Lots of other features
- 

The systemd units

- **Unit** configuration files live under `/lib/systemd/system/` (Debian) or `/usr/lib/systemd/system/`
 - **Enabled** (on startup) and custom unit configuration files are placed under `/etc/systemd/system/`
 - Unit types:
 - `<name>.service`: for services
 - `<name>.socket`: for IPC sockets or FIFO buffers
 - `<name>.device`: for systemd managed devices
 - `<name>.mount`: for systemd managed mount points
 - `<name>.target`: for boot targets (similar to runlevels)
 - Lots of other unit types:
 - `.automount`, `.swap`, `.path`, `.timer`, `.snapshot`, `.slice`, `.scope`
- 

An example *systemd* service file

- `$ cat /lib/systemd/system/myservice.service`

```
[Unit]
```

```
Description=MyService Description # Service Description  
After=postgresql.service # start after postgresql service
```

```
[Service]
```

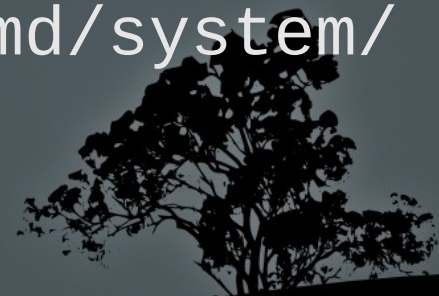
```
EnvironmentFile=-/etc/default/myservice # read environment  
# vars from here  
WorkingDirectory=/srv/myservice # service working directory  
ExecStart=/usr/sbin/myservice -r $OPTS # the cli command to  
# run our service  
KillMode=process # kill only the main process on stop  
Restart=on-failure # Restart the services if it fails or  
# crashes
```

```
[Install]
```

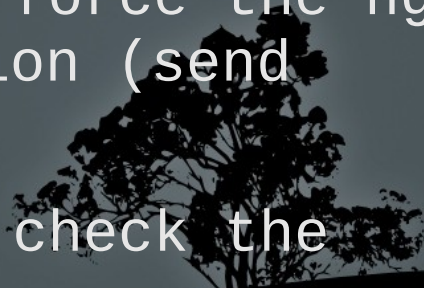
```
WantedBy=multi-user.target # this service is called my the  
# multi-user target (equivalent  
# to the multi-user runlevel of  
# sysvinit
```

The `systemctl` command

- `systemd` behavior is controlled with the `systemctl` command
- `$ systemctl list-units --type=service #`
list all `systemd` services
- `$ systemctl list-units --type=service --`
`state=running #` list all running services
- `$ systemctl list-units --type=target #` list
all targets
- `$ systemctl daemon-reload #` reload `systemd`
after a unit file configuration change or
a new file under `/etc/systemd/system/`



The `systemctl` command

- `$ systemctl enable nginx.service # enable the nginx service on startup`
 - `$ systemctl disable nginx.service # disable the nginx service from startup`
 - `$ systemctl start nginx.service # start the nginx service`
 - `$ systemctl stop nginx.service # stop the nginx service`
 - `$ systemctl restart nginx.service # restart the nginx service`
 - `$ systemctl reload nginx.service # force the nginx service to reload its configuration (send SIGHUP)`
 - `$ systemctl status nginx.service # check the status of the nginx service`
- 

The `shutdown` command

- `# shutdown -h now # initiate system poweroff without delay`
- `# shutdown -r now # initiate system restart without delay`
- `# shutdown -h +10 Please log out now! # initiate system poweroff in 10 minutes and notify all system users`
- `# shutdown -r 3:00 # restart at 3:00 in the morning`



The `shutdown` command

Options:

- **-h** # system halt or poweroff
- **-r** # system restart
- **-k** # send warning but without halt or restart
- **-f** # skip filesystem check (fsck)
- **-F** # force filesystem check (fsck)
- **-t 2** # 2 seconds delay between warning and sending SIGKILL to processes



Sending messages with `wall`

- **wall** (warn all) is a utility for sending messages to all open terminals in a system
- `$ wall "This is the end!"` # send the quoted message to all active terminals in the system

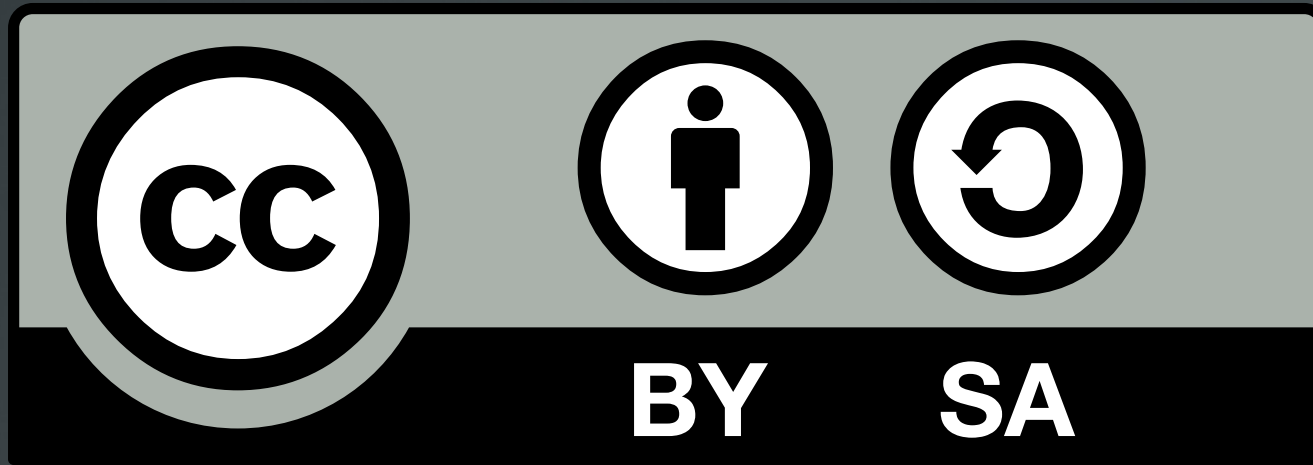


The `poweroff`, `halt` and `reboot` commands

- `# poweroff # power system off without delay`
- `# halt # halt system without delay`
- `# reboot # reboot system without delay`



License



The work titled "LPIC-1 101-500 – Lesson 11" by Theodotos Andreou is distributed with the Creative Commons Attribution ShareAlike 4.0 International License.

