Command Equivalents - SysVinit and systemd

Action	SysVinit	systemd
Start/stop/restart/reload/status of a service	service ntpd [start | stop | etc...]	systemctl [start | stop etc...] ntpd.service
Restart a service only if already running	service ntpd condrestart	systemctl condrestart httpd.service
Enable or disable service on startup	chkconfig ntpd [off | on]	systemctl [enable | disable] ntpd.service
Is service enabled at startup (this runlevel)?	chkconfig ntpd	systemctl is-enabled ntpd.service
List services that can be started or stopped Used to list all the services and other units	ls /etc/rc.d/init.d/	systemctl OR systemctl list-unit-files --type=service OR ls /lib/systemd/system/*.service AND ls /etc/systemd/system/*.service
Print table of services listing runlevels each is configured on or off	chkconfig --list	systemctl list-unit-files --type=service ls /etc/systemd/system/*.wants/
Print a table of services that will be started when booting into graphical mode	chkconfig --list | grep 5:on	systemctl list-dependencies graphical.target
List what levels this service is config'd on/ off	chkconfig ntpd --list	ls /etc/systemd/system/*.wants/ntpd.service
Create a new service file or modify config	chkconfig ntpd --add	systemctl daemon-reload (this reloads systemd!)
Suspend the system	pm-suspend	systemctl suspend
Hibernate	pm-hibernate	systemctl hibernate
Follow the system log file	tail -f /var/log/messages (or /var/log/syslog)	journalctl -f
System halt	telinit 0, poweroff, halt	systemctl isolate poweroff.target | systemctl poweroff
Change to Single-user mode	telinit 1, s, single	systemctl isolate rescue.target (or runlevel1.target)
Change to Multi-user	telinit 2	systemctl isolate multi-user.target (or runlevel2.target*)
Change to Multi-user with Network	telinit 3	systemctl isolate multi-user.target (or runlevel3.target)
Change to RunLevel 4	telinit 4	systemctl isolate multi-user.target (or runlevel4.target*)
Change to Multi-user, w/ network, x11	telinit 5	systemctl isolate graphical.target (or runlevel5.target)
Reboot	telinit 6, reboot	systemctl isolate reboot.target | systemctl reboot
Emergency Shell	init emergency	emergency.target
Check current runlevel	runlevel	runlevel (deprecated) OR systemctl | **grep (script)**
Change default runlevel	sed s/^id:.*:initdefault:/id:3:initdefault:/	systemctl set-default multi-user.target
Set multi-user target on next boot	sed s/^id:.*:initdefault:/id:3:initdefault:/	ln -sf /lib/systemd/system/multi-user.target /etc/systemd/system/default.target
Execute a systemd cmd on remote host		systemctl dummy.service start -H user@host
Check boot time		systemd-analyze or systemd-analyze time
Kill all processes related to a service		systemctl kill dummy
Get logs for events for today		journalctl --since=today
Hostname and other host information		hostnamectl
Date and time		timedatectl

All recent versions of systemctl assume ".service" if left off. So, 'systemctl start myservicename.service' works like 'systemctl start myservicename'
Default systemd Fedora installs; 0, 1, 3, 5, and 6; have a 1:1 mapping with a specific systemd *target*.
** If you use runlevels 2 or 4 it is suggested that you make a new named systemd *target* as /etc/systemd/system/$YOURTARGET that takes one of the existing runlevels as a base (you can look at /lib/systemd/system/graphical.target as an example), make a directory /etc/systemd/system/$YOURTARGET.wants, and then symlink the additional services that you want to enable into that directory.
Runlevels 2 and 4 are by default just "multi-user" runlevel3 in systemd until defined otherwise

systemd Command Overview

Is systemd is installed on the system? Is it running?	**# systemd-run --version -- # ps -eaf | grep [s]ystemd**
List all the available units . [*.service, *.mount, *.socket, *.device]	**# systemctl list-unit-files**
List all running units. [*.service, *.mount, *.socket, *.device]	**# systemctl list-units**
List all failed units . [*.service, *.mount, *.socket, *.device]	**# systemctl --failed**
Analyze the systemd boot process.	**# systemd-analyze**
Analyze time taken by each process at boot.	**# systemd-analyze blame**
Analyze critical chain at boot (all or a specific service, etc)	**# systemd-analyze critical-chain** (OR **critical-chain httpd.service**)
	"@" = Time after unit is active or started. "+" = Time unit takes to start

Using systemd to Manage Mountpoints, Sockets, Devices Just Like Services

The general systemctl commands that work with services also do the same thing for mountpoints, sockets, and devices (which are seen as service types). Simply specify in the place of "name.service" the proper item, such as tmp.mount, cups.socket, or item.device. The list-unit-files option uses the **--type** directive such as **--type=device** or **--type=socket** accordingly. Starting stopping a mount point simply mounts and unmounts the mountpoint [**systemctl list-unit-files --type=mount** will list all mountpoints, for example]

# systemctl list-unit-files --type=socket
# systemctl [start | restart | stop | reload | status | is-active | enable | disable | is-enabled | mask | unmask] tmp.mount

How to enable, disable or check if turned on at boot time (auto start)	**# systemctl [is-active | enable | disable] httpd.service**
How to mask (making it impossible to start) or unmask a service	**# systemctl [mask | unmask] httpd.service**
List all services (including enabled and disabled).	**# systemctl list-unit-files --type=service**
Start, restart, stop, reload and check the status of a service	**# systemctl [start | restart | stop | reload | status] httpd.service**
Check all the configuration details of a service	**# systemctl show httpd**
Get a list of dependencies for a service	**# systemctl list-dependencies httpd.service**
How to a Kill a service using systemctl command.	**# systemctl kill httpd**
Is unit enabled or not right now ("is-active" is for the target's config)	**# systemctl is-enabled crond.service**
Get current CPU Shares of a Service (default CPUShare = 1024)	**# systemctl show -p CPUShares httpd.service**
Increase/decrease CPU share of a process	**# systemctl set-property httpd.service CPUShares=2000**
No unit specified means default.target - Requires=, RequiresOverridable=, Requisite=, RequisiteOverridable=, Wants=, BindsTo= dependencies	
List control groups hierarchically	**# systemd-cgls**
List control groups according to CPU, memory, Input and Output	**# systemd-cgtop**

To enable a service, you must be currently running the target you want the service to start in.
For example, to turn on bluetooth.service in the graphical.target, you have to change to the graphical.target first with **isolate**, then run **enable**.
systemctl isolate graphical.target ; **systemctl enable bluetooth.service**
Makes a symlink **/etc/systemd/system/graphical.target.wants/bluetooth.servic**e pointing to **/usr/lib/systemd/system/bluetooth.service**

How to start system rescue mode	**# systemctl rescue**
How to enter into emergency mode.	**# systemctl emergency**
List current default runlevel in use.	**# systemctl get-default**
Start Runlevel 5 aka graphical mode	**# systemctl isolate runlevel5.target** (OR **graphical.target**)
Set multiuser mode (runlevel 3) as default	**# systemctl set-default runlevel3.target** (OR **multiuser.target**)
*[This set-default line creates a symlink **/etc/systemd/system/default.target** pointing to **/usr/lib/systemd/system/multiuser.target**]*	
Reboot, halt, suspend, hibernate or put system in hybrid-sleep	**# systemctl [reboot | halt | suspend | hibernate | hybrid-sleep]**

Unit and target files in **/usr/lib/systemd/system/** are pointed to by symlinks placed in **/etc/systemd/system/**
Unit files enabled for a specific target will have a symlink in that target's "wants" directory, such as **/etc/systemd/system/multi-user.target.wants**

Runlevel Service Management Tools -SysVinit initscript utilities

service *<servicename>* [**start** | **stop** | **restart** | **status** | **list**] Activate, etc., a daemon in current runlevel
OR Go to **/etc/rc.d/init.d/** directory and type ***./<servicename>* start**.
init OR **telinit [0-6]** switches to the specified runlevel
runlevel tells you your runlevel- returns two numbers - 3 5 means that current runlevel is 5 and previous was 3.
chkconfig --list gives you this type of output:

```
[root@localhost rc.d]# chkconfig --list
NetworkManager  0:off   1:off   2:off   3:off   4:off   5:off   6:off
acpid           0:off   1:off   2:on    3:on    4:on    5:on    6:off
anacron         0:off   1:off   2:on    3:on    4:on    5:on    6:off
apmd            0:off   1:off   2:on    3:on    4:on    5:on    6:off
atd             0:off   1:off   2:off   3:on    4:on    5:on    6:off
auditd          0:off   1:off   2:on    3:on    4:on    5:on    6:off
autofs          0:off   1:off   2:off   3:on    4:on    5:on    6:off
avahi-daemon    0:off   1:off   2:off   3:on    4:on    5:on    6:off
```

chkconfig --list *<servicename>* will output the same on one line for that one service
chkconfig --list | grep *<servicename>* might be more helpful to list multiple matches (e.g. "avahi-" services)
chkconfig --level 35 *<servicename>* off | **on** | **reset** | **resetpriorities** - affects service in runlevels 3 and 4
chkconfig --level *<servicename>* on -turns service on in levels 2-5, if 'off' affects 0-6
chkconfig --add OR **--del** adds or removes scripts from **/etc/rc.d/init.d/**
chkconfig --override *<servicename>* - files in **/etc/chkconfig.d/*<servicename>*** can override init service scripts

ntsysv is just a Red Hat TUI interface to turn off and on services in the currently active runlevel. Debian has **rcconf**
ntsysv --runlevel 35 (OR **--level 35**) manages services on levels 3 and 5
redhat-config-services or **system-config-services** - graphical Services Configuration Tool

/sbin/telinit is linked to /sbin/init - takes a one-character argument (0-6 for switching runlevels, s/S/1 all work for single user mode, U/u to resart current runlevel init scrips without checking inittab; Q/q to do so forcing checking inittab. The init binary checks if it is init or telinit by looking at its process id; the real init's process id is always 1.

- ls /etc/init.d/ will also list all of the currently available service files
- Scripts to stop and start processes can be used as an alternative to running **kill**.
- neither ntsysv or chkconfig starts or stops services- only dictates runlevel. The service command does that.
- xinetd services are immediately affected by ntsysv, unlike others

SysVinit Runlevel	***Systemd Target***	***Description***
0	*poweroff.target*	*Halts the system*
1	*rescue.target*	*Single-user mode* (everything mounted, minimal services)
2	*multi-user.target*	*Multiuser mode without networking*
3	*multi-user.target*	*Multiuser mode with networking*
4	*multi-user.target*	*User configurable*
5	*graphical.target*	*Used for the GUI (X11 multiuser mode)*
6	*reboot.target*	*Reboots the system*

systemd's emergency.target
- Is like init=/bin/sh on the kernel command line
- Has no corresponding sysvinit runlevel- would just boot your machine to a shell with really nothing started.
- You get a shell, but almost nothing else (except for systemd in the background)
- No services are started, no mount points mounted, no sockets established.
- Useful for running specific scripts which could then be started independently.
- Allows booting bit-by-bit, starting the various services and other units step-by-step manually.

In SysVinit, services can define arbitrary commands. Examples would be service iptables panic, or service httpd graceful. Native systemd services do not have this ability. Any service that defines an additional command in this way would need to define some other, service-specific, way to accomplish this task when writing a native systemd service definition.Check the package-specific release notes for any services that may have done this.

SysVinit - Directory Structures

Generally, you will find in the **/etc** directory some symlinks to stuff that is actually in the **/etc/rc.d/** directory. This can cause some confusion, since we have some other symlink stuff for backward compatibility for systems that once supported Upstart but no longer do so. This writing ignores all of that and sticks to CentOS 5.5

/etc/init.d is a symlink to the directory **/etc/rc.d/init.d** and the same with **/etc/rc#.d** linking to **/etc/rc.d/rc#.d**, also the same with scripts **rc**, **rc.local** and **rc.sysinit**, who's actual locations is also in the /etc/rc.d/ directory as well
Even though you will often see these in **/etc.** Here is where they actually live:

```
/etc/rc.d/init.d/
/etc/rc.d/rc0.d/
/etc/rc.d/rc1.d/
```

...and so on....

```
/etc/rc.d/rc5.d/
/etc/rc.d/rc6.d/
/etc/rc.d/rc
/etc/rc.d/rc.local
/etc/rc.d/rc.sysinit
```

Service's scripts are in /etc/rc.d/init.d/ (often accessible via the symlink **/etc/init.d/**)
- Each service managed by SystemVinit needs a script in /etc/rc.d/init.d/
- Common elements to the scripts in **/etc/rc.d/init.d/*<servicename>*** are the top several lines, beginning in a prelude declaring the script processor (as in **#!/bin/bash**); followed by a line with name and brief description; another with chkconfig default runlevels the service should be started, and the start and stop priority levels.
- If default is to not be started in any runlevels, a "-" should be used in place of the runlevels list.
- Another entry contains service description (used by ntsysv)
- Finally the general functions container for init.d scripts is defined(usually **/etc/init.d/functions**), followed by lines setting and ENVVARS and functions for the service, (much like in bashrc does for it's purpose).

For example, the beginning of **/etc/rc.d/init.d/kudzu** has these line common to SysVinit scripts:

```
#!/bin/bash
# kudzu          This scripts runs the kudzu hardware probe.
# chkconfig: 345 05 95
# description:   This runs the hardware probe, and optionally configures \
#                changed hardware.
# Source function library.
. /etc/init.d/functions
```

Says that the script should be started in levels 3, 4, and 5, start priority 5, stop priority 95

/etc/rc.d/init.d directory contents:

```
/etc/rc.d/init.d/acpid
/etc/rc.d/init.d/anacron
```

...

```
/etc/rc.d/init.d/ypbind
/etc/rc.d/init.d/yum-updatesd
```

Files in the**/etc/rc.d/rc#.d** directories are symlinks to the actual scripts for all of SysVinit's managed programs in **/etc/rc.d/init.d** For example, **/etc/rc.d/rc0.d/K99cpuspeed** links to **/etc/rc.d/init.d/cpuspeed**
With those links, the naming convention of K or S means "kill" or "start" and the number (like 99) indicates the numerical order that it is executed in that runlevel's directory, when that runlevel starts. This way, it is directed that things are stopped and started in the proper order.

As an example, here is a sample of some filenames in **/etc/rc.d/rc3.d/**

```
K88wpa_supplicant
K89netplugd
K89rdisc
K91capi
K99readahead_later
S00microcode_ctl
S02lvm2-monitor
S04readahead_early
S05kudzu
S08ip6tables
S08iptables
S08mcstrans
```

rc.local is to execute commands during the startup without needing symlinks. "Local system initialization script"

S99local -> softlink for /etc/rc.local in 2,3,4 and 5 runlevels

You can optionally have a similar shutdown items script in **/etc/rc.d/rc.local_shutdown**

rc.sysinit seems to be redhat specific and is executed very early in the process while rc.local is executed later.

rc is typically not used by linux distributions but is used in BSD

The rc stands for "run commands"; runcom (as in .cshrc or /etc/rc) comes from the runcom facility from the MIT CTSS system, ca. 1965. From Kernighan and Ritchie, as told to Vicki Brown: "There was a facility that would execute a bunch of commands stored in a file; it was called runcom for "run commands", and the file began to be called "a runcom". rc in Unix is a fossil from that usage."

The idea of having the command processing shell be an ordinary slave program came from the Multics design, and a predecessor program on CTSS by Louis Pouzin called RUNCOM. The first time I remember the name "shell" for this function was in a Multics design document by Doug Eastwood (of BTL). Commands that return a value into the command line were called "evaluated commands" in the original Multics shell, which used square brackets where Unix uses backticks.

(/etc/inittab   Main config file for SysVinit)

- Specifies runlevels, scripts to run when certain runlevels are selected, and items to respawn (getty).
- Syntax for an entry in inittab: id:runlevels:action:process.
- First is a unique arbitrary identifier, second indicates what runlevels invoke the command, third is how to handle this entry (like execute command once or respawn whenever it exits, fourth is the command and it's arguments
- x:5:respawn:/etc/X11/prefdm -nodaemon   Runlevel 5, specify default login screen for X11
- 3:2345:respawn:/sbin/mingetty tty3   Virtual terminal 3, available for runlevels 2 through 5

Sample /etc/initab (truncated):

```
# Set the default runlevel to three - points to line below "l3:3:wait:/etc/rc.d/rc 3"
id:3:initdefault:

# Execute /etc/rc.d/rc.sysinit when the system boots
# starts network, establishes mounted systems, starts SELinux, encryption
si:S:sysinit:/etc/rc.d/rc.sysinit

# Run /etc/rc.d/rc with the runlevel as an argument - e.g., a 5 points it to /etc/rc5.d/
# Runlevels are designated in /etc/rc.d/
l0:0:wait:/etc/rc.d/rc 0
l1:1:wait:/etc/rc.d/rc 1
...
l5:5:wait:/etc/rc.d/rc 5
l6:6:wait:/etc/rc.d/rc 6

# Executed when we press ctrl-alt-delete
ca::ctrlaltdel:/sbin/shutdown -t3 -rf now

# Start agetty for virtual consoles 1 through 6
c1:12345:respawn:/sbin/agetty 38400 tty1
c2:12345:respawn:/sbin/agetty 38400 tty2
...
c6:45:respawn:/sbin/agetty 38400 tty6
```

Default runlevel is determined, then scripts in appropriate **/etc/rc.d/rcX/** directory are run.

When SysVinit is instructed to change runlevels, it reads initab for what **/etc/rc.d** directory belongs to that runlevel.

The **rc.d** directory contains the daemon scripts which run at boot and when switching runlevels.

Contents in **/etc/rc.d/rcX/** directories are just symlinks to the files in **/etc/rc.d/init.d/** and are named to either start with an S (start) or a K (kill), in order of the number to be processed

For example: take a symlink **S45dhcpd** in in **/etc/rc.d/rc3/** - This means the **/etc/rc.d/init.d/dhcpd** script will be 45th in order to start for that runlevel directory containing this symlink- in this case runlevel 3.

Some types of Linux using SysVinit don't even use this system of symlinks. Slackware uses something similar to BSD where all directives for a runlevel are only put in a runlevel script.

Example systemd Unit Script Contents

It is *not* advised to edit unit scripts in **/usr/lib/systemd/system/** so they remain default/as-installed by packages.

For custom changes, make copy to edit in **/etc/systemd/system** - this directory overrides those defaults for you.

Example Service Unit Script - /usr/lib/systemd/system/httpd.service contents:

```
[Unit]
Description=The Apache HTTP Server
After= network.target remote-fs.target nss-lookup.target

[Service]
Type=notify
EnvironmentFile=/etc/sysconfig/httpd
ExecStart=/usr/sbin/httpd $OPTIONS -DFOREGROUND
ExecReload=/usr/sbin/httpd $OPTIONS -k graceful
ExecStop=/bin/kill -WINCH ${MAINPID}
KillSignal=SIGCONT
PrivateTmp=true

[Install]
WantedBy=multi-user.target
```

Example Target Unit Script - /usr/lib/systemd/system/multi-user.target contents:

```
[Unit]
Description= Multi-User System
Documentation=man:systemd.special(7)
Requires=basic.target
Conflicts=rescue.service rescue.target
After=basic.target rescue.service rescue.target
AllowIsolate=yes
[Install]
Alias=default.target
```

"After" is what is loaded after this finishes activating

"Requires" should be what needs to be loaded before

Example Custom Mount Scripts - /etc/systemd/system/lvdisk.mount and lvdisk.automount

Needs a mount file and automount file with a matching name (mydisk5.automount for mydisk5.mount)

This is the way future versions of RHEL will likely do automount instead of the old /etc/fstab method

```
        # vim /etc/systemd/system/lvdisk.mount
[Unit]
Description= Example test mount
[Mount]
what  = /dev/vgdisk/lvdisk
where = /lvdisk
type = xfs
[Install]
WantedBy=multi-user.target

        # vim /etc/systemd/system/lvdisk.automount
[Unit]
Description= Example test automount
[Automount]
where = /lvdisk
[Install]
WantedBy=multi-user.target
```

Test it out - systemctl enable lvdisk.automount; systemctl start lvdisk.automount; mount | grep lvdisk

Contents of the systemd Package

Installed programs: bootctl, busctl, coredumpctl, halt, hostnamectl, init, journalctl, kernel-install, localectl, loginctl, machinectl, networkctl, poweroff, reboot, runlevel, shutdown, systemctl, systemd-analyze, systemd-ask-password, systemd-cat, systemd-cgls, systemd-cgtop, systemd-delta, systemd-detect-virt, systemd-escape, systemd-hwdb, systemd-inhibit, systemd-machine-id-setup, systemd-mount, systemd-notify, systemd-nspawn, systemd-path, systemd-resolve, systemd-run, systemd-socket-activate, systemd-stdio-bridge, systemd-tmpfiles, systemd-tty-ask-password-agent, telinit, timedatectl, and udevadm

Installed libraries: libnss_myhostname.so.2, libnss_mymachines.so.2, libnss_resolve.so.2, libnss_systemd.so.2, libsystemd.so, libsystemd-shared-231.so, and libudev.so

Installed directories: /etc/binfmt.d, /etc/init.d, /etc/kernel, /etc/modules-load.d, /etc/sysctl.d, /etc/systemd, /etc/tmpfiles.d, /etc/udev, /etc/xdg/systemd, /lib/systemd, /lib/udev, /usr/include/systemd, /usr/lib/binfmt.d, /usr/lib/kernel, /usr/lib/modules-load.d, /usr/lib/sysctl.d, /usr/lib/systemd, /usr/lib/tmpfiles.d, /usr/share/doc/systemd-234, /usr/share/factory, /usr/share/systemd, /var/lib/systemd, and /var/log/journal

bootctl	Query the firmware and boot manager settings
busctl	Review logs and monitor the D-Bus bus
coredumpctl	Retrieve coredumps from the systemd Journal
halt	Normally invokes **shutdown** with the *-h* option, except when already in run-level 0, then it tells the kernel to halt the system; it notes in the file /var/log/wtmp that the system is being brought down
hostnamectl	Query and change the system hostname and related settings
init	The first process to be started when the kernel has initialized the hardware which takes over the boot process and starts all the proceses it is instructed to
journalctl	Query the contents of the systemd Journal
kernel-install	Add and remove kernel and initramfs images to and from /boot
localectl	Query and change the system locale and keyboard layout settings
loginctl	Review logs and control the state of the systemd Login Manager
machinectl	Review logs and control the state of the systemd Virtual Machine and Container Registration Manager
networkctl	Review logs and state of the network links as seen by systemd-networkd
poweroff	Tells the kernel to halt the system and switch off the computer (see **halt**)
reboot	Tells the kernel to reboot the system (see **halt**)
runlevel	Reports the previous and the current run-level, as noted in the last run-level record in /var/run/utmp
shutdown	Brings the system down in a secure way, signaling all processes and notifying all logged-in users
systemctl	Review logs and control the state of the systemd system and service manager
systemd-analyze	Determine system boot-up performance of the current boot
systemd-ask-password	Query a system password or passphrase from the user, using a question message specified on the command line
systemd-cat	Connect STDOUT and STDERR of a process with the Journal
systemd-cgls	Recursively shows the contents of the selected Linux control group hierarchy in a tree
systemd-cgtop	Shows the top control groups of the local Linux control group hierarchy, ordered by their CPU, memory and disk I/O load
systemd-delta	Identify and compare configuration files in /etc that override default counterparts in /usr
systemd-detect-virt	Detects execution in a virtualized environment
systemd-escape	Escape strings for inclusion in systemd unit names
systemd-hwdb	Manage hardware database (hwdb)
systemd-inhibit	Execute a program with a shutdown, sleep or idle inhibitor lock taken
systemd-machine-id-setup	Used by system installer tools to initialize the machine ID stored in /etc/machine-id at install time with a randomly generated ID
systemd-mount	A tool to temporarily mount or auto-mount a drive.
systemd-notify	Used by daemon scripts to notify the init system about status changes
systemd-nspawn	Run a command or OS in a light-weight namespace container
systemd-path	Query system and user paths
systemd-resolve	Resolve domain names, IPV4 and IPv6 addresses, DNS resource records, and services
systemd-run	Create and start a transient .service or a .scope unit and run the specified command in it
systemd-socket-activate	A tool to listen on socket devices and launch a process upon connection.
systemd-tmpfiles	Creates, deletes and cleans up volatile and temporary files and directories, based on the configuration file format and location specified in tmpfiles.d directories
systemd-tty-ask-password-agent	Used to list or process pending systemd password requests
telinit	Tells **init** which run-level to change to
timedatectl	Query and change the system clock and its settings
udevadm	Generic Udev administration tool: controls the udevd daemon, provides info from the Udev database, monitors uevents, waits for uevents to finish, tests Udev configuration, and triggers uevents for a given device
libsystemd	systemd utility library
libudev	A library to access Udev device information

-- from http://www.linuxfromscratch.org/lfs/view/systemd/chapter06/systemd.html

You may have noticed that installed items listed contains telinit, and an /etc/init.d directory. According to the Fedora Wiki, the 'service' and 'chkconfig' commands will (surprisingly) mostly continue to work as expected in the systemd world. Presumably, this would be for backward compatibility support for old scripts, etc.

Target unit directories hold symlinks to the real unit files like this:	**/etc/systemd/system/XXXXXX.target.wants/bluetooth.service**
Those symlinks point to the actual service (etc) unit files that reside here:	**/usr/lib/systemd/system/bluetooth.service**
The **default.target** file here is a target/runlevel symlink:	**/etc/systemd/system/default.target**
And the default.target symlink points to the actual target here:	**/usr/lib/systemd/system/XXXXXX.target**

Running **systemctl isolate graphical.target** will not affect the default.target symlink, and merely switches the current runlevel (use **set-default**).

Running **systemctl disable myservice** basically does the same as **rm '/etc/systemd/system/multi-user.target.wants/service.myservice'**
Running **systemctl enable myservice** basically does the same as
ln -s '/usr/lib/systemd/system/myservice.service' '/etc/systemd/system/multi-user.target.wants/service.myservice'

The "**target.wants**" directories in **/usr/lib/systemd/system/** hold symlinks to the corresponding runlevel's unit files just like init's **/etc/rc.d/rc#.d/**
A target is itself a unit file, manages other unit files. Defaults are multi-user.target, graphical.target, rescue.target, emergency.target, poweroff.target, and reboot.target