**SysVinit to Systemd Cheatsheet**

This is a document to help system administrators who need to understand what commands in systemd replace their old workflow in sysvinit. If you want general information on systemd, refer to [systemd](https://fedoraproject.org/wiki/Systemd).

[Idea.png](https://fedoraproject.org/wiki/File:Idea.png)

**Note on 'service' and 'chkconfig' commands**   
The 'service' and 'chkconfig' commands will mostly continue to work as expected in the systemd world, this guide is how to use the native systemctl replacements.

**Services**

Note that all recent versions of systemctl assume the '.service' if left off. So, 'systemctl start frobozz.service' is the same as 'systemctl start frobozz'

|  |  |  |
| --- | --- | --- |
| **Sysvinit Command** | **Systemd Command** | **Notes** |
| service frobozz start | systemctl start frobozz | Used to start a service (not reboot persistent) |
| service frobozz stop | systemctl stop frobozz | Used to stop a service (not reboot persistent) |
| service frobozz restart | systemctl restart frobozz | Used to stop and then start a service |
| service frobozz reload | systemctl reload frobozz | When supported, reloads the config file without interrupting pending operations. |
| service frobozz condrestart | systemctl condrestart frobozz | Restarts if the service is already running. |
| service frobozz status | systemctl status frobozz | Tells whether a service is currently running. |
| ls /etc/rc.d/init.d/ | systemctl (or) systemctl list-unit-files --type=service (or)  ls /lib/systemd/system/\*.service /etc/systemd/system/\*.service | Used to list the services that can be started or stopped  Used to list all the services and other units |
| chkconfig frobozz on | systemctl enable frobozz | Turn the service on, for start at next boot, or other trigger. |
| chkconfig frobozz off | systemctl disable frobozz | Turn the service off for the next reboot, or any other trigger. |
| chkconfig frobozz | systemctl is-enabled frobozz | Used to check whether a service is configured to start or not in the current environment. |
| chkconfig --list | systemctl list-unit-files --type=service (or) ls /etc/systemd/system/\*.wants/ | Print a table of services that lists which runlevels each is configured on or off |
| chkconfig frobozz --list | ls /etc/systemd/system/\*.wants/frobozz.service | Used to list what levels this service is configured on or off |
| chkconfig frobozz --add | systemctl daemon-reload | Used when you create a new service file or modify any configuration |

Note that all /sbin/service and /sbin/chkconfig lines listed above continue to work on systemd, and will be translated to native equivalents as necessary. The only exception is chkconfig --list.

[Warning.png](https://fedoraproject.org/wiki/File:Warning.png)

**Additional commands**  
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.

**Runlevels/targets**

Systemd has a concept of *targets* which serve a similar purpose as runlevels but act a little different. Each *target* is named instead of numbered and is intended to serve a specific purpose. Some *targets* are implemented by inheriting all of the services of another *target* and adding additional services to it. There are systemd *target*s that mimic the common sysvinit runlevels so you can still switch *target*s using the familiar telinit RUNLEVEL command. The runlevels that are assigned a specific purpose on vanilla Fedora installs; 0, 1, 3, 5, and 6; have a 1:1 mapping with a specific systemd *target*. Unfortunately, there's no good way to do the same for the user-defined runlevels like 2 and 4. If you make use of those 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. (The service unit files that you symlink live in /lib/systemd/system).

|  |  |  |
| --- | --- | --- |
| **Sysvinit Runlevel** | **Systemd Target** | **Notes** |
| 0 | runlevel0.target, poweroff.target | Halt the system. |
| 1, s, single | runlevel1.target, rescue.target | Single user mode. |
| 2, 4 | runlevel2.target, runlevel4.target, multi-user.target | User-defined/Site-specific runlevels. By default, identical to 3. |
| 3 | runlevel3.target, multi-user.target | Multi-user, non-graphical. Users can usually login via multiple consoles or via the network. |
| 5 | runlevel5.target, graphical.target | Multi-user, graphical. Usually has all the services of runlevel 3 plus a graphical login. |
| 6 | runlevel6.target, reboot.target | Reboot |
| emergency | emergency.target | Emergency shell |

Changing runlevels:

|  |  |  |
| --- | --- | --- |
| **Sysvinit Command** | **Systemd Command** | **Notes** |
| telinit 3 or init 3 | systemctl isolate multi-user.target (OR systemctl isolate runlevel3.target OR telinit 3) | Change to multi-user run level.  (Change temporarily) |
| sed s/^id:.\*:initdefault:/id:3:initdefault:/ | ln -sf /lib/systemd/system/multi-user.target /etc/systemd/system/default.target  (lib/systemd/system/multi-user.target or /usr/lib/systemd/system/multi-user.target) | Set to use multi-user runlevel on next reboot.  (Change permanently) |

Kernel Options:

The above systemd targets can be used when booting. At the GRUB menu, edit the selection to add "systemd.unit=*target*" (without the double-quotation marks) as a kernel option where *target* is one of the above. (For example, "rescue.target".)

Tip: the ".target" extention is optional. The "systemd.unit=rescue" kernel option works the same as "systemd.unit=rescue.target".

s: symbolic

f: force