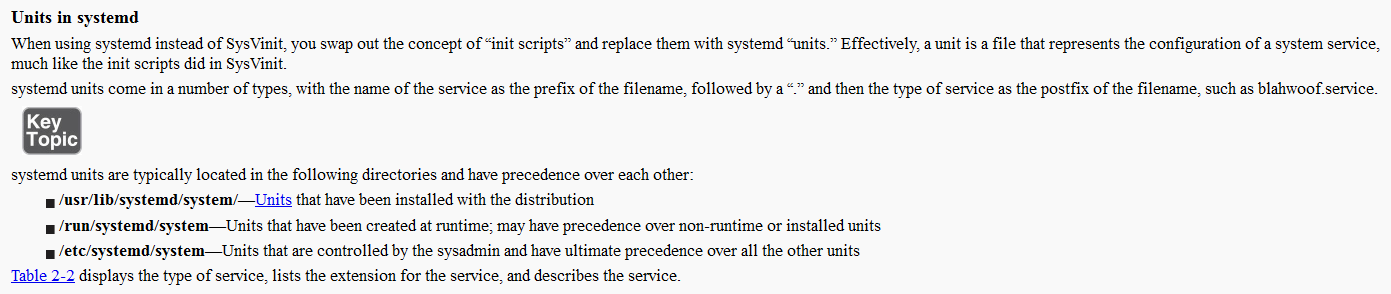
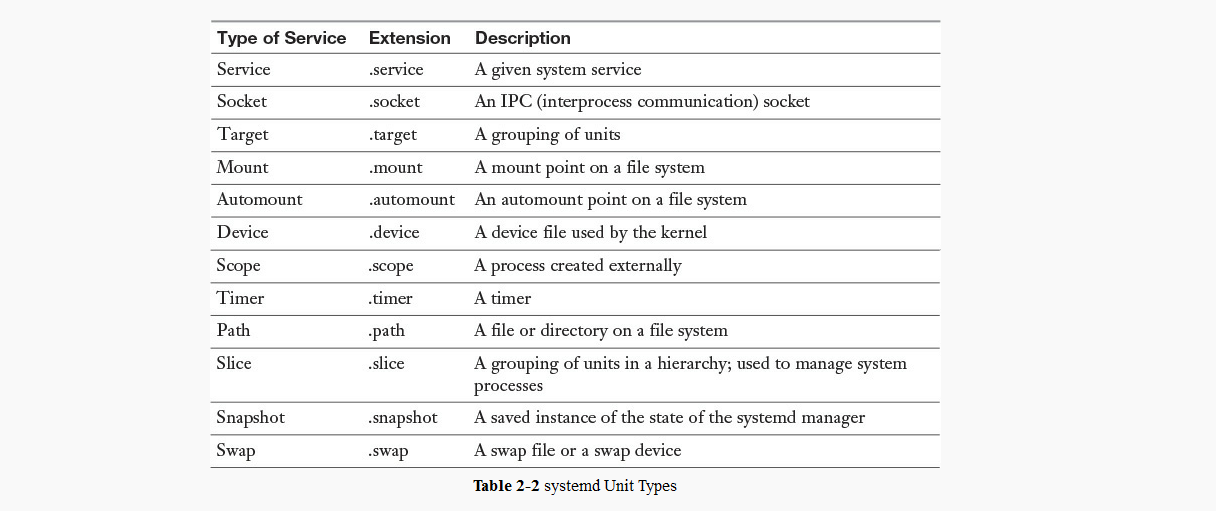
/etc/system/system > /run/system/system > /usr/lib/systemd/system

Units are the objects that systemd knows how to manage. These are basically a standardized representation of system resources that can be managed by the suite of daemons and manipulated by the provided utilities.





**Wants and Requires**

The systemd procedure uses a set of options or requirements called “wants” and “requires” to indicate dependencies between units and groups of units.

For example, if you were to look at the typical contents of the /usr/lib/system/system/graphical.target file, you might find the following lines:

Requires=multi-user.target  
After=multi-user.target  
Wants=display-manager.service

The Requires statement means that before the graphical.target can be processed, all the items listed in the multi-user.target must be started, which means that all the linked services in the /usr/lib/systemd/system/multi-user.target directory must be started.

Only after that requirement is met will the Wants statement be processed (which again means the starting of linked services in the appropriate directory, this time the /usr/lib/systemd/sytem/multi-user.target.wants directory).

After=multi-user.target => /usr/lib/system/system/graphical.target is started after starting multi-user.target . multi-user.target is started first, then graphical.target is started

Before=, After=

A space-separated list of unit names. Configures ordering dependencies between units. If a unit foo.service contains a setting Before=bar.service and both units are being started, bar.service's start-up is delayed until foo.service is started up. Note that this setting is independent of and orthogonal to the requirement dependencies as configured by Requires=. It is a common pattern to include a unit name in both the After= and Requires= option, in which case the unit listed will be started before the unit that is configured with these options. This option may be specified more than once, in which case ordering dependencies for all listed names are created. After= is the inverse of Before=, i.e. while After= ensures that the configured unit is started after the listed unit finished starting up, Before= ensures the opposite, i.e. that the configured unit is fully started up before the listed unit is started. Note that when two units with an ordering dependency between them are shut down, the inverse of the start-up order is applied. i.e. if a unit is configured with After= on another unit, the former is stopped before the latter if both are shut down. Given two units with any ordering dependency between them, if one unit is shut down and the other is started up, the shutdown is ordered before the start-up. It doesn't matter if the ordering dependency is After= or Before=. It also doesn't matter which of the two is shut down, as long as one is shut down and the other is started up. The shutdown is ordered before the start-up in all cases. If two units have no ordering dependencies between them, they are shut down or started up simultaneously, and no ordering takes place.

Requires=

Configures requirement dependencies on other units. If this unit gets activated, the units listed here will be activated as well. If one of the other units gets deactivated or its activation fails, this unit will be deactivated. This option may be specified more than once or multiple space-separated units may be specified in one option in which case requirement dependencies for all listed names will be created. Note that requirement dependencies do not influence the order in which services are started or stopped. This has to be configured independently with the After= or Before= options. If a unit foo.service requires a unit bar.service as configured with Requires= and no ordering is configured with After= or Before=, then both units will be started simultaneously and without any delay between them if foo.service is activated. Often, it is a better choice to use Wants= instead of Requires= in order to achieve a system that is more robust when dealing with failing services.

Note that dependencies of this type may also be configured outside of the unit configuration file by adding a symlink to a .requires/ directory accompanying the unit file. For details, see above.