Name

bootup — System bootup process

Description

A number of different components are involved in the boot of a Linux system. Immediately after power-up, the system firmware will do minimal hardware initialization, and hand control over to a boot loader (e.g. systemd-boot(7) or GRUB) stored on a persistent storage device. This boot loader will then invoke an OS kernel from disk (or the network). On systems using EFI or other types of firmware, this firmware may also load the kernel directly.

The kernel (optionally) mounts an in-memory file system, often generated by dracut(8), which looks for the root file system. Nowadays this is implemented as an "initramfs" — a compressed CPIO archive that the kernel extracts into a tmpfs. In the past normal file systems using an in-memory block device (ramdisk) were used, and the name "initrd" is still used to describe both concepts. It's the boot loader or the firmware that loads both the kernel and initrd/initramfs images into memory, but the kernel which interprets it as a file system. systemd(1) may be used to manage services in the initrd, similarly to the real system.

After the root file system is found and mounted, the initrd hands over control to the host's system manager (such as systemd(1)) stored in the root file system, which is then responsible for probing all remaining hardware, mounting all necessary file systems and spawning all configured services.

On shutdown, the system manager stops all services, unmounts all file systems (detaching the storage technologies backing them), and then (optionally) jumps back into the initrd code which unmounts/detaches the root file system and the storage it resides on. As a last step, the system is powered down.

Additional information about the system boot process may be found in boot(7).

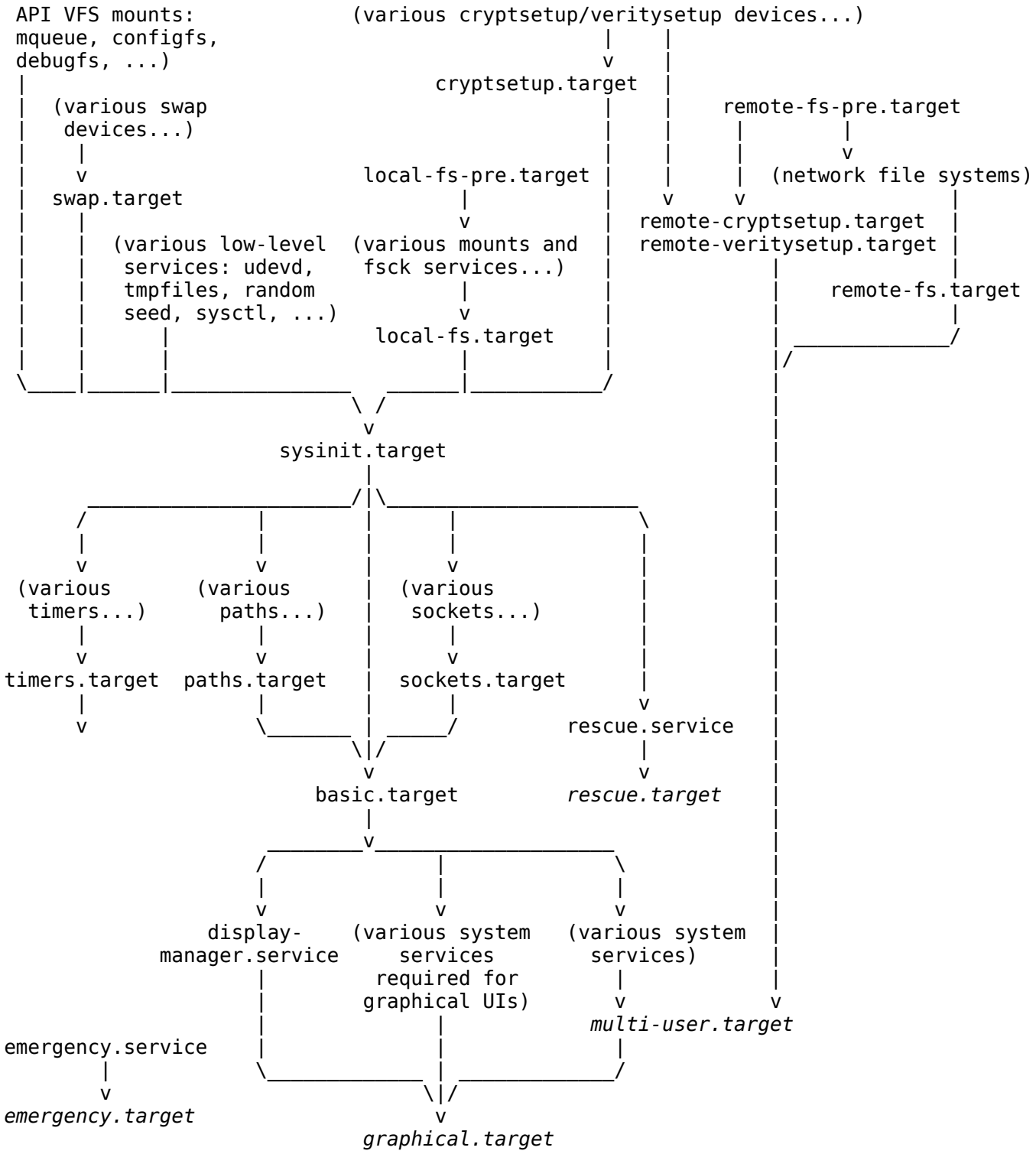
System Manager Bootup

At boot, the system manager on the OS image is responsible for initializing the required file systems, services and drivers that are necessary for operation of the system. On systemd(1) systems, this process is split up in various discrete steps which are exposed as target units. (See systemd.target(5) for detailed information about target units.) The boot-up process is highly parallelized so that the order in which specific target units are reached is not deterministic, but still adheres to a limited amount of ordering structure.

When systemd starts up the system, it will activate all units that are dependencies of `default.target` (as well as recursively all dependencies of these dependencies). Usually, `default.target` is simply an alias of `graphical.target` or `multi-user.target`, depending on whether the system is configured for a graphical UI or only for a text console. To enforce minimal ordering between the units pulled in, a number of well-known target units are available, as listed on systemd.special(7).

The following chart is a structural overview of these well-known units and their position in the boot-up logic. The arrows describe which units are pulled in and ordered before which other units. Units near the top are started before units nearer to the bottom of the chart.

```
                          cryptsetup-pre.target veritysetup-pre.target
                                                |
(various low-level                              v
```



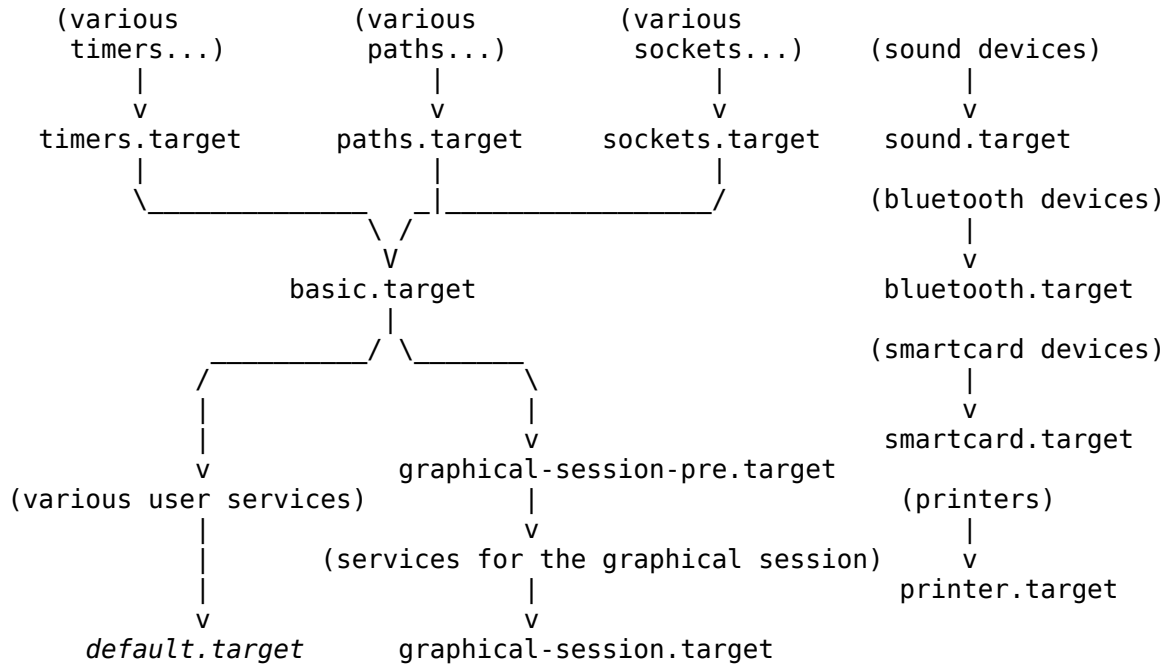
Target units that are commonly used as boot targets are *emphasized*. These units are good choices as goal targets, for example by passing them to the `systemd.unit=` kernel command line option (see systemd(1)) or by symlinking `default.target` to them.

`timers.target` is pulled-in by `basic.target` asynchronously. This allows timers units to depend on services which become only available later in boot.

User manager startup

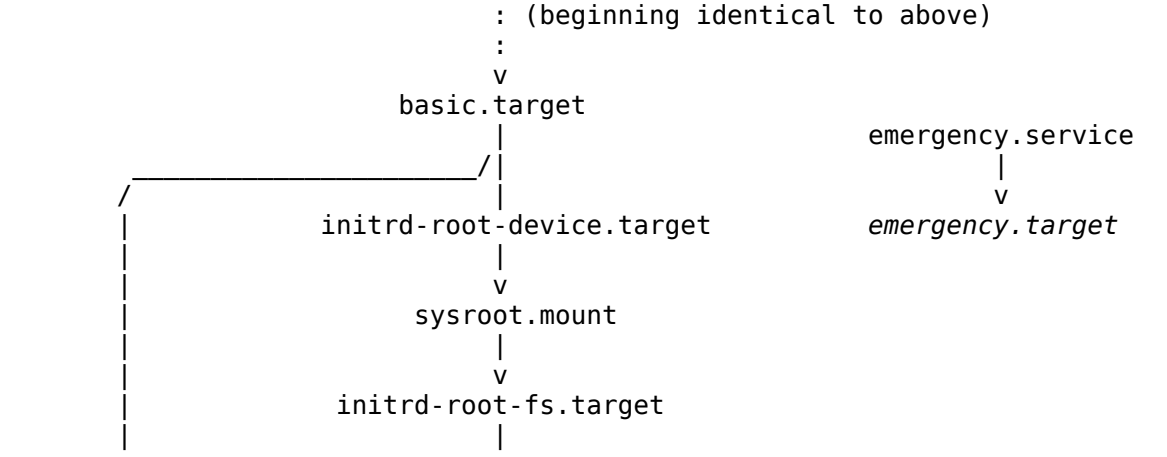
The system manager starts the `user@`*`uid`*`.service` unit for each user, which launches a separate unprivileged instance of **systemd** for each user — the user manager. Similarly to the system manager, the user manager starts

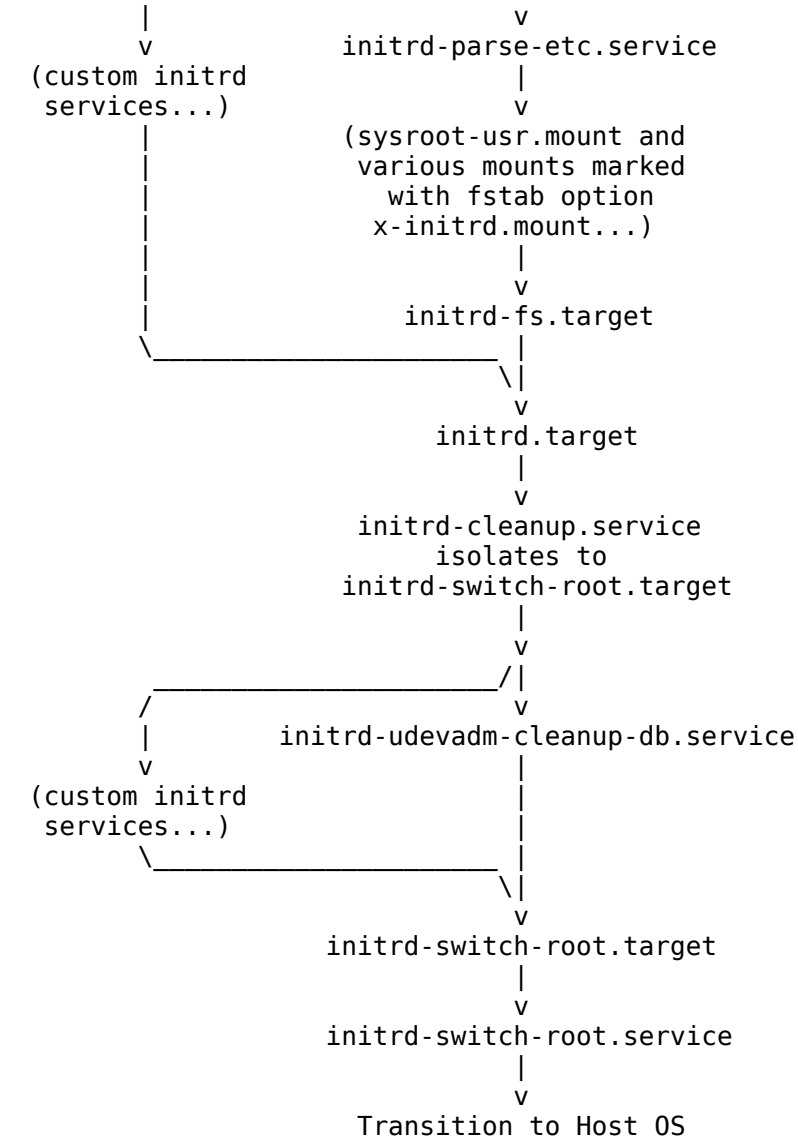
units which are pulled in by `default.target`. The following chart is a structural overview of the well-known user units. For non-graphical sessions, `default.target` is used. Whenever the user logs into a graphical session, the login manager will start the `graphical-session.target` target that is used to pull in units required for the graphical session. A number of targets (shown on the right side) are started when specific hardware is available to the user.



Bootup in the initrd

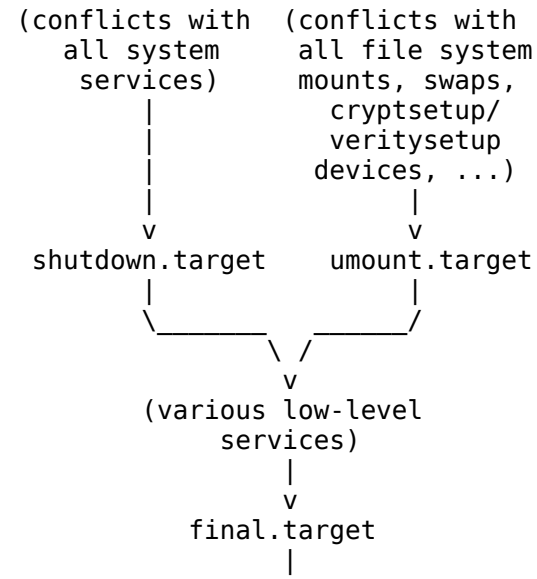
Systemd can be used in the initrd as well. It detects the initrd environment by checking for the `/etc/initrd-release` file. The default target in the initrd is `initrd.target`. The bootup process is identical to the system manager bootup until the target `basic.target`. After that, systemd executes the special target `initrd.target`. Before any file systems are mounted, the manager will determine whether the system shall resume from hibernation or proceed with normal boot. This is accomplished by `systemd-hibernate-resume@.service` which must be finished before `local-fs-pre.target`, so no filesystems can be mounted before the check is complete. When the root device becomes available, `initrd-root-device.target` is reached. If the root device can be mounted at `/sysroot`, the `sysroot.mount` unit becomes active and `initrd-root-fs.target` is reached. The service `initrd-parse-etc.service` scans `/sysroot/etc/fstab` for a possible `/usr/` mount point and additional entries marked with the *x-initrd.mount* option. All entries found are mounted below `/sysroot`, and `initrd-fs.target` is reached. The service `initrd-cleanup.service` isolates to the `initrd-switch-root.target`, where cleanup services can run. As the very last step, the `initrd-switch-root.service` is activated, which will cause the system to switch its root to `/sysroot`.

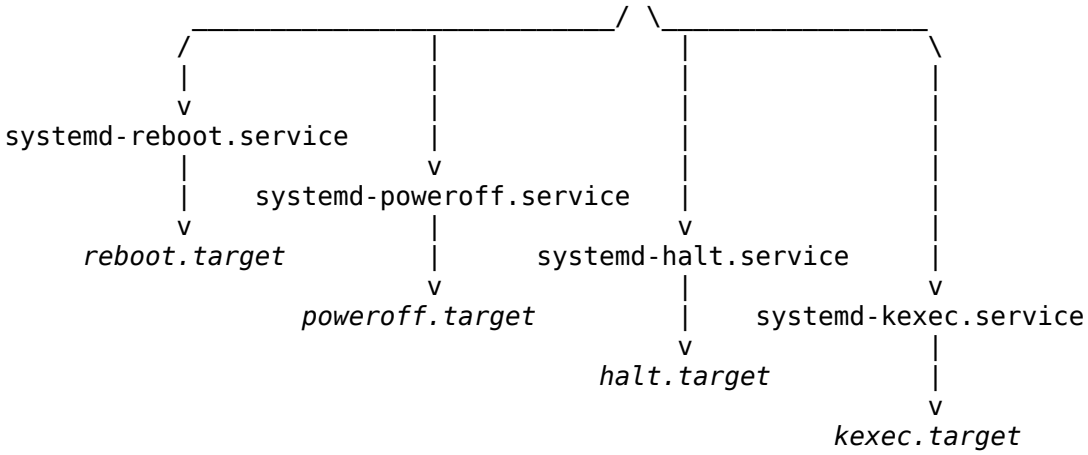




System Manager Shutdown

System shutdown with systemd also consists of various target units with some minimal ordering structure applied:





Commonly used system shutdown targets are *emphasized*.

Note that systemd-halt.service(8), `systemd-reboot.service`, `systemd-poweroff.service` and `systemd-kexec.service` will transition the system and server manager (PID 1) into the second phase of system shutdown (implemented in the `systemd-shutdown` binary), which will unmount any remaining file systems, kill any remaining processes and release any other remaining resources, in a simple and robust fashion, without taking any service or unit concept into account anymore. At that point, regular applications and resources are generally terminated and released already, the second phase hence operates only as safety net for everything that couldn't be stopped or released for some reason during the primary, unit-based shutdown phase described above.

See Also

systemd(1), boot(7), systemd.special(7), systemd.target(5), systemd-halt.service(8), dracut(8)