Startup and Shutdown

Run the code in the boot ROM (BIOS or UEFI).

x86 Boot: The code in the boot rom does the following. There is a boot order in the BIOS, attempt to boot from the various devices in the order given until one works (Newer systems may use UEFI instead of BIOS but the concept is the same).

Disk boot:
1) Read the Master Boot Record into RAM.
2) Run the code you just loaded.
The MBR contains the Linux loader (LILO or GRUB)
This loads the bootstrap program which loads the kernel plus its startup code, using a list of disk sectors.

The bootstrap program loads these sectors and branches to the startup code.

The start up code decompresses the kernel,
probes the hardware, and loads the correct drivers.

Then it starts the first process (`init`)

`init` starts the system daemons,
configures network (if enabled)
starts the login programs (`agetty`)

Init

This is always process number 1, treat it carefully.

`/etc/inittab` – controls init behavior

Format : delimited fields (like passwd)
First field is an identifier or capability,
second field is a list of runlevels
third field is an action
—(`wait` init should wait for completion)
—(`sysinit` do at boot (before `boot` and `bootwait`)
fourth field is a process or command to run
—`/sbin/agetty`

`kill -HUP 1`
reread the `inittab` file,
`1` is the process ID of the init process

At boot time `init` takes actions directed by script files
found in the `/etc/rc.d` directory
`inittab` controls the use of these files
(says which files to use and which to ignore)

Linux: The main multiuser boot file is `/etc/rc.d/rc.M`,
it invokes many the other files involved in multiuser boot.

Run-level: Linux has run levels
–multiuser: full capabilities
–single user: used for basic maintenance

`initdefault` in `inittab` specifies default run level

`init` commands specify the run levels to which they apply.

`telinit`: tell `init` to change run levels.
`/sbin/telinit 1`: enter run-level 1 (single user)
level 3 is multi user

Booting to single user may be specified at the `Boot:` prompt.

Shutdown: `init` warns all running processes by sending them a `SIGTERM`. After a wait, all processes that haven't voluntarily quit are forcibly terminated with a `SIGKILL`.

Logging: each time a process stops, `init` records the information in `utmp` and `wtmp`.

`last`: summarize `wtmp`

Shutdown

The file system is cached and live.
Daemons are active even when no user is logged in.

You must flush the cache and close the file system to ensure integrity.

Do not power-off or reset

`lab17# shutdown -h now`
`shutdown`: Halt the daemons, flush the cache, halt the system. (Actually `init` does all the work.)
`lab17# reboot`
`reboot`: Halt the daemons, flush the cache, reboot the system.
equivalent: `shutdown -r now`
equivalent: `ctl-alt-del` (maybe, see `inittab`)

A disorderly shutdown requires a full consistency check of the file system (fsck).

An orderly shutdown results in a clean file system, minimal consistency check, and quick boot/reboot.

System V.4 Boot

The is a directory for each run level
level 1: `/etc/rc1.d`
level 3: `/etc/rc3.d`

In each of these directory there are script files starting with `S` if they are to be used during startup and `K` if they to be used during shutdown. Usually, these are just soft links to real files in `/etc/init.d`

Example:

```
S30inet
S34route
S42nfs
```

The initializations are run in numeric order, inet first, route next, nfs last.

`S` files are run with a start parameter:

```
S34route start
```

`K` files are run with a stop parameter

Systemd

Replaces `init` on many modern systems.

Does all of the startup and shutdown work that init does and much MUCH more.

Config directory is `/etc/systemd/system/`

Still mostly uses symlinks.

`systemctl` command controls various functionality.

For example:
`systemctl start ssh` starts the sshd service.

`systemctl stop ssh` stops the sshd service.

Uses unit files instead of script files.

Example systemd unit file:

```
[Unit]
Description=OpenBSD Secure Shell server
After=network.target auditd.service
ConditionPathExists=!/etc/ssh/sshd_not_to_be_run
[Service]
EnvironmentFile=-/etc/default/ssh
ExecStartPre=/usr/sbin/sshd -t
ExecStart=/usr/sbin/sshd -D $SSHD_OPTS
ExecReload=/usr/sbin/sshd -t
ExecReload=/bin/kill -HUP $MAINPID
KillMode=process
Restart=on-failure
RestartPreventExitStatus=255
Type=notify
[Install]
WantedBy=multi-user.target
Alias=sshd.service
```