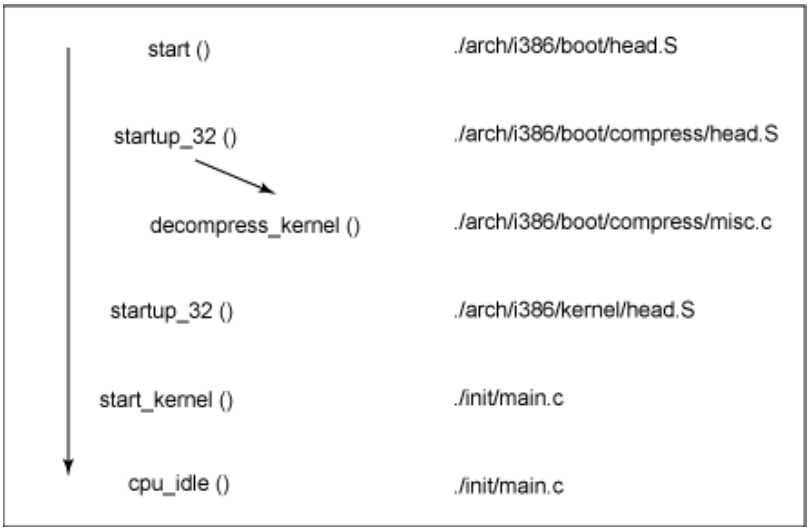
Init

Linux kernel boot



init_post code

```
static int noinline init_post(void)
{
    free_initmem();
    unlock_kernel();
    mark_rodata_ro();
    system_state = SYSTEM_RUNNING;
    numa_default_policy();

    if  ( sys_open((const char __user *) "/dev/console", O_RDWR, 0) < 0 )
          printk(KERN_WARNING "Warning: unable to open an initial console.\n");

    (void) sys_dup(0);
    (void) sys_dup(0);

    if  ( ramdisk_execute_command ) {
          run_init_process(ramdisk_execute_command);
          printk(KERN_WARNING "Failed to execute %s\n",
                    ramdisk_execute_command);
    }

    /*
     * We try each of these until one succeeds.
     *
     * The Bourne shell can be used instead of init if we are
     * trying to recover a really broken machine.
     */
    if  ( execute_command  ) {
          run_init_process(execute_command);
          printk(KERN_WARNING "Failed to execute %s.  Attempting "
                        "defaults...\n", execute_command);
    }
    run_init_process("/sbin/init");
    run_init_process("/etc/init");
    run_init_process("/bin/init");
    run_init_process("/bin/sh");

    panic("No init found.  Try passing init= option to kernel.");
}
```

Source:
init/main.c
in Linux 2.6.22

Init process

- The first thing the kernel does is to execute **init** program
- **Init** runs with PID=1
- **Init** is the parent of all processes executing on Linux
- **Init** reads configuration from /etc/inittab
- The first process that **init** starts is /etc/rc.d/rc.sysinit
- Based on the appropriate run-level, scripts are executed to start various processes
- Init typically will start multiple instances of "getty" which waits for console logins
- Upon shutdown, init controls the sequence and processes for shutdown

inittab file

- The inittab file defines the initial runlevel and what to do in each runlevel
- { Identifier:RunLevels:Action:Command }

```
id:5:initdefault:                    → Defines the initial runlevel
si::sysinit:/etc/init.d/rcS          → Runs the system startup script
l0:0:wait:/etc/init.d/rc 0
l1:1:wait:/etc/init.d/rc 1
l2:2:wait:/etc/init.d/rc 2
l3:3:wait:/etc/init.d/rc 3
l4:4:wait:/etc/init.d/rc 4
l5:5:wait:/etc/init.d/rc 5           → Runs (and waits for) rc 5
1:2345:respawn:/sbin/getty 38400 tty1 → Keeps getty tty1 running
2:23:respawn:/sbin/getty 38400 tty2
3:23:respawn:/sbin/getty 38400 tty3
4:23:respawn:/sbin/getty 38400 tty4
```


Runlevels

- A runlevel is a software configuration of the system which allows only a selected group of processes to exist
- The processes spawned by init for each of these runlevels are defined in the /etc/inittab file
- Init can be in one of eight runlevels: 0-6

Runlevel	Scripts Directory	State
0	/etc/rc.d/rc0.d/	shutdown/halt system
1	/etc/rc.d/rc1.d/	Single user mode
2	/etc/rc.d/rc2.d/	Multiuser with no network services exported
3	/etc/rc.d/rc3.d/	Default text/console only start. Full multiuser
4	/etc/rc.d/rc4.d/	Reserved for local use. Also X-windows (Slackware/BSD)
5	/etc/rc.d/rc5.d/	XDM X-windows GUI mode (Redhat/System V)
6	/etc/rc.d/rc6.d/	Reboot
s or S		Single user/Maintenance mode (Slackware)
M		Multiuser mode (Slackware)

Changing runlevel at boot time

- LILO: append the runlevel to the boot command :
  - LILO: linux 3 or
  - LILO: linux 5

- GRUB: press the `e' key to edit the configuration
  - append the runlevel to the end of the boot command as shown:
  - kernel /vmlinuz ro root=/dev/hda1 **5**

/etc/init.d/ files

```
#!/bin/sh
#
# chkconfig: 345 91 35
# description: Starts / stops the atd daemon

# Source function library.
. /etc/rc.d/init.d/functions

# Check that Server file exists.
[ -f $SERVER ] || exit 0

# See how we were called.
case "$1" in
  start)
        echo -n "Starting atd services: “
        daemon  /sbin/atd
        echo_success
        touch /var/lock/subsys/atd
        ;;
  stop)
        echo -n "Shutting down atd services: "
        killproc atd -TERM
        rm -f /var/lock/subsys/atd
        echo ""
        ;;
  status)
        status atd
        ;;
  restart)
        echo -n "Restarting atd services: "
        $0 stop
        $0 start
        echo "done."
        ;;
  *)
        echo "Usage: atd {start|stop|restart|status}"
        exit 1
esac
```

init.d

- Deamon is a background process
- init.d is a directory that admin can use to start/stop individual demons
  - /etc/rc.d/init.d/ (Red Hat/Fedora)
  - /etc/init.d/ (Suse)
  - /etc/init.d/ (Debian)
- Admin can issuing the command and one option:
  - start, stop, status, restart or reload
- i.e. to stop the web server:
  - /etc/init.d/httpd stop
  - service httpd stop
  - systemctl stop httpd

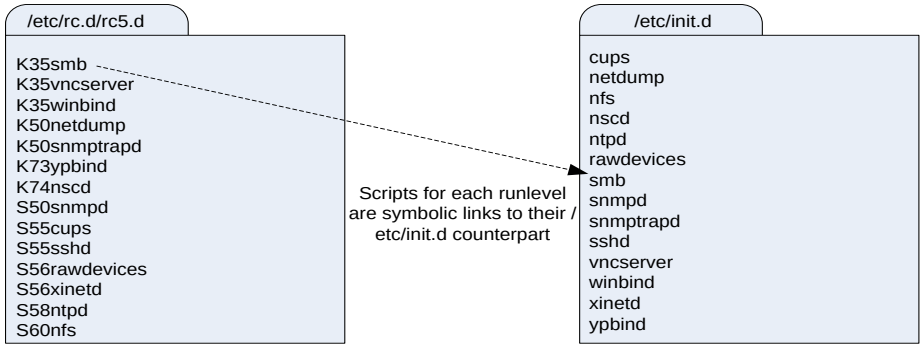
/etc/rc

```
# New run level
runlevel=$1

# First, run the KILL scripts.
for i in /etc/rc$runlevel.d/K* ; do
      $i  stop
done

# Now run the START scripts.
for i in /etc/rc$runlevel.d/S* ; do
      $i start
done
```

```
$ ls /etc/rc4.d
K25squid -> ../init.d/squid
K35dhcpd -> ../init.d/dhcpd
S26ntpd -> ../init.d/ntpd
S30x10 -> ../init.d/x10
S55sshd -> ../init.d/sshd
S60lpd -> ../init.d/lpd
S78mysqld -> ../init.d/mysqld
S80sendmail -> ../init.d/sendmail
S90crond -> ../init.d/crond
S97httpd -> ../init.d/httpd
S98smb -> ../init.d/smb
```

rc#.d/ scripts

- The system first runs the scripts whose names start with K to kill the associated processes → /etc/rc.d/init.d/<command> stop
- The system runs the scripts whose names start with S to start the processes → /etc/rc.d/init.d/<command> start



K = Kill
S = Start

chkconfig

- *chkconfig* command
  - *Allows to turn services on, off for one or more runlevels*
  - *Allows to add, delete services for one or more runlevels*

 # chkconfig --list

 # chkconfig --level 2345 ssh on

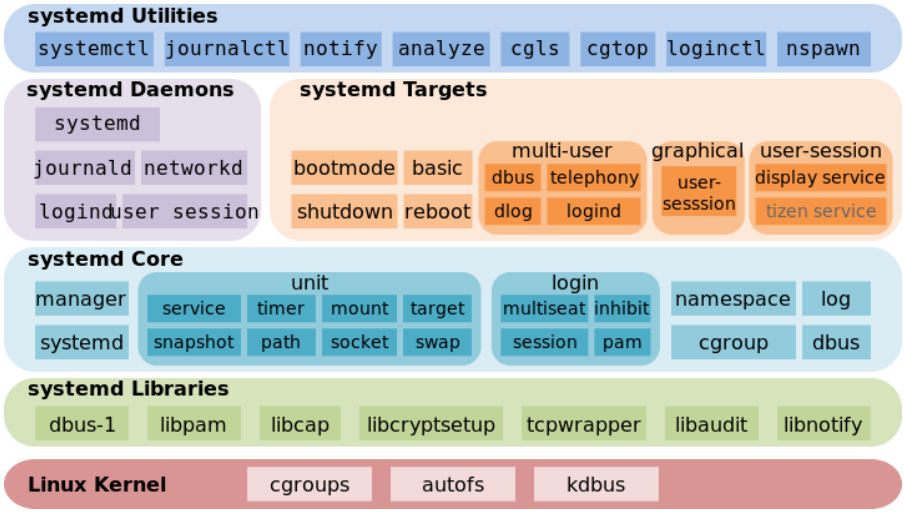
 # chkconfig --add ssh

Systemd

- Becoming more prevalent in Linux Distros
  - Currently used by Fedora and OpenSUSE
  - Testing in Debian

- Mostly compatible with the init (SystemV) system
  - init scripts are read as alternative format
  - Established functionality such as fstab still supported

Systemd



Systemd style

- No runlevels. ->Targets
- No /etc/inittab file,
- No shell scripts for startup,
- No /etc/init.d/ directory

- Everything is in /etc/systemd/system/ with lots of symlinks to files in /usr/lib/systemd/system/ which are the startup “scripts”
- Main commands:
  - systemctl for controlling most everything
  - journalctl for logs and errors

Targets - runlevels

- Systemd defines targets instead of runlevels.
- Standard targets:
  - poweroff.target (runlevel 0)
  - rescue.target (runlevel 1)
  - Multi-user.target (runlevel 2,3 en 4)
  - Graphical.target (runlevel 5)
  - Reboot.target (runlevel 6)
- Runlevels may still exist as symbolic links to new targets.

Systemd - Units

- Works on “units”
- Uses a dependency system between “units”
  - Requires/Wants
  - Conflicts
  - Before
  - After
- Encapsulate objects relevant to booting and maintenance
- Configured in config files (unit files)
- May be tied through symbolic links

Systemd – Unit Types

- Different unit types
  - service: handles daemons
  - socket: handles network sockets
  - target: Logical grouping of units (example: runlevel)
  - device: expose kernel devices
  - mount: controls mount points of the files system
  - automount: mounts the file system
  - snapshot: references other units (similar to targets)

systemctl

- Most common Units are services
- To start or stop a service
  - systemctl start blah.service
  - systemctl stop blah.service
- Restart and reload
  - systemctl restart blah.service
  - systemctl try-restart blah.service
  - systemctl reload blah.service

systemctl

- Enabling or disabling units
  - systemctl enable blah.service
  - systemctl disable blah.service
- This is tied into “targets” (run levels).
  - So the blah will start when the system is “booted” to that target.
  - systemctl list-units shows the status of all the units for the current target.
  - systemctl list-units --all shows all units for all targets.
- See all the units and if they are enabled or not
  - systemctl list-unit-files

Logging and info

- When a unit starts up or fails it shows no information
  - Instead we need to use the logging.
  - The most basic is journalctl
    - It will show almost everything from boot.
  - journalctl –k will show only the kernel message, which is like the old dmesg command.

Systemd Unit File Section

- [Unit]
  - Description
  - Requires
  - Wants
  - Conflicts
  - Before
  - After

Systemd Service Section

- [Service]
  - Type=

    simple|oneshot|forking|dbus|notify|idle
  - ExecStart
  - ExecReload
  - ExecStop
  - Restart=

    no|on-success|on-failure|on-abort|always

sshd.service

```
[Unit]
Description=OpenSSH server daemon
Documentation=man:sshd(8) man:sshd_config(5)
After=network.target sshd-keygen.target
Wants=sshd-keygen.target

[Service]
Type=notify
EnvironmentFile=-/etc/sysconfig/sshd-permitrootlogin
EnvironmentFile=-/etc/sysconfig/sshd
ExecStart=/usr/sbin/sshd -D $OPTIONS $PERMITROOTLOGIN
ExecReload=/bin/kill -HUP $MAINPID
KillMode=process
Restart=on-failure
RestartSec=42s

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

sshd.socket

[Unit]

Description=OpenSSH Server Socket

Documentation=man:sshd(8) man:sshd_config(5)

Conflicts=sshd.service

[Socket]

ListenStream=22

Accept=yes

[Install]

WantedBy=sockets.target

Mount via systemd

- .mount file in /etc/systemd/system

```
# cat opt-lvdisk1.mount
[Unit]
Description=mount lvdisk1 on /opt/lvdisk1

[Mount]
What=/dev/vgroup/lvdisk1
Where=/opt/lvdisk1
Type=xfs

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

Systemctl mount:

- systemctl start opt-lvdisk1.mount