Today: OS Boot Process

Powering On a PC

- Six step process for a generic OS boot
  - many steps before the kernel starts booting

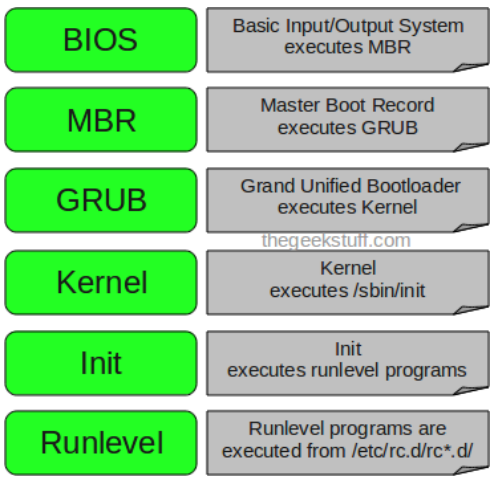
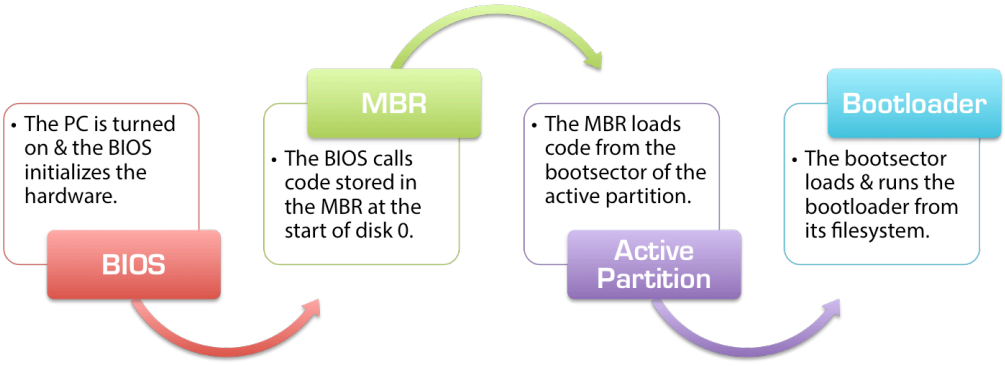


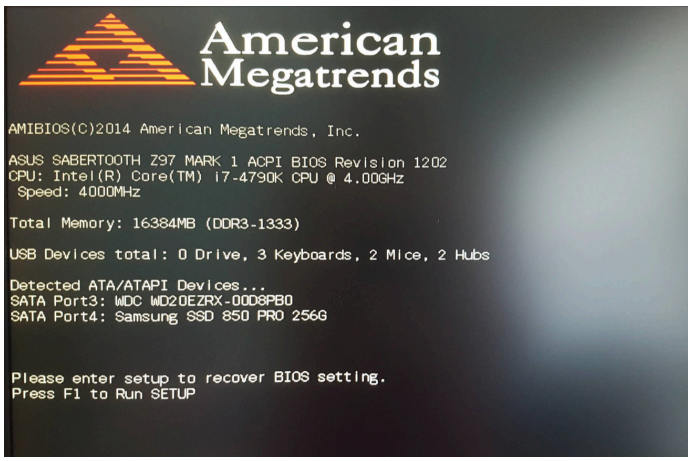
figure: courtesy thegeekstuff.com

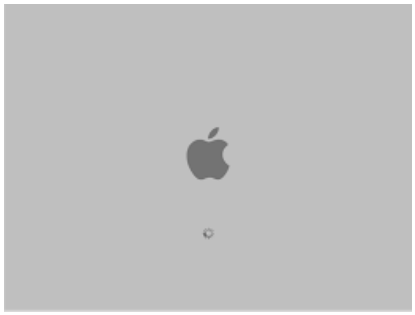
BIOS/MBR Boot

- BIOS: basic input output system
- MBR: Master Boot Record on disk
  - Refer to https://neosmart.net/wiki/mbr-boot-process/
  -



BIOS Splash Screen



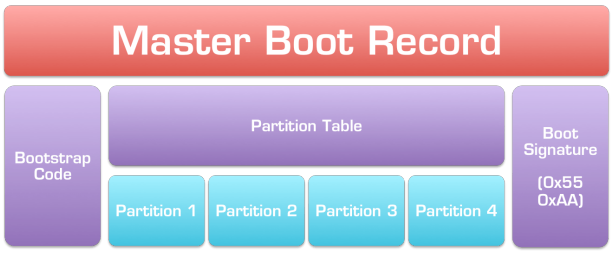


BIOS

- BIOS: basic input output system
  - lowest level of software on a PC that interfaces with hardware
  - Stored on EEPROM / ROM
  - interface for boot loader or kernel to communicate with and control hardware
    - Use interrupts to read/write to desk and other hardware
- Power on: Power-on Self Test
  - Init RAM, scan for attached hardware (disks), USB devices, quick tests performed (E.g., keyboard error), inits video card
- BIOS then does a boot handoff
  - Loads a small program from boot device/disk
  - Hand off control to the program

Boot Device and MBR Boot

- Master boot record: first sector of boot device/disk
- Partition table: unto 4 partitions
  - Default: entire disk has one partition; multiple partitions: each can be a different volume/file system
- Bootstrap code: called stage 1 boot loader
  - First 440 bytes of 512 bytes
  - CPU loads bootstrap and starts executing it
  - Look for partition marked active; load code from that partition



MBR and Partition Table Entry

Structure of a classical generic MBR

Address		Description		Size (bytes)
Hex	Dec			
000_{hex}	+0	Bootstrap code area		446
$1BE_{hex}$	+446	Partition entry №1	*Partition table* (for primary partitions)	16
$1CE_{hex}$	+462	Partition entry №2		16
$1DE_{hex}$	+478	Partition entry №3		16
$1EE_{hex}$	+494	Partition entry №4		16
$1FE_{hex}$	+510	55_{hex}	*Boot signature*[a]	2
$1FF_{hex}$	+511	AA_{hex}		
Total size: 446 + 4×16 + 2				**512**

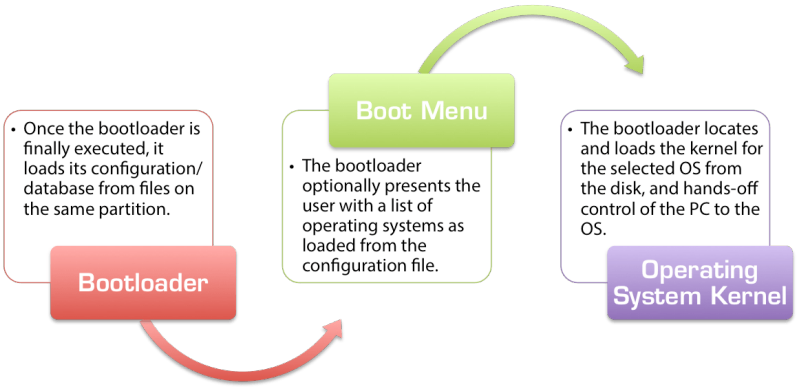
Element (offset)	Size	Description
0	byte	Boot indicator bit flag: 0 = no, 0x80 = bootable (or "active")
1	byte	Starting head
2	6 bits	Starting sector (Bits 6-7 are the upper two bits for the Starting Cylinder field.)
3	10 bits	Starting Cylinder
4	byte	System ID
5	byte	Ending Head
6	6 bits	Ending Sector (Bits 6-7 are the upper two bits for the ending cylinder field)
7	10 bits	Ending Cylinder
8	uint32_t	Relative Sector (to start of partition -- also equals the partition's starting LBA value)
12	uint32_t	Total Sectors in partition

Partition Boot Sector

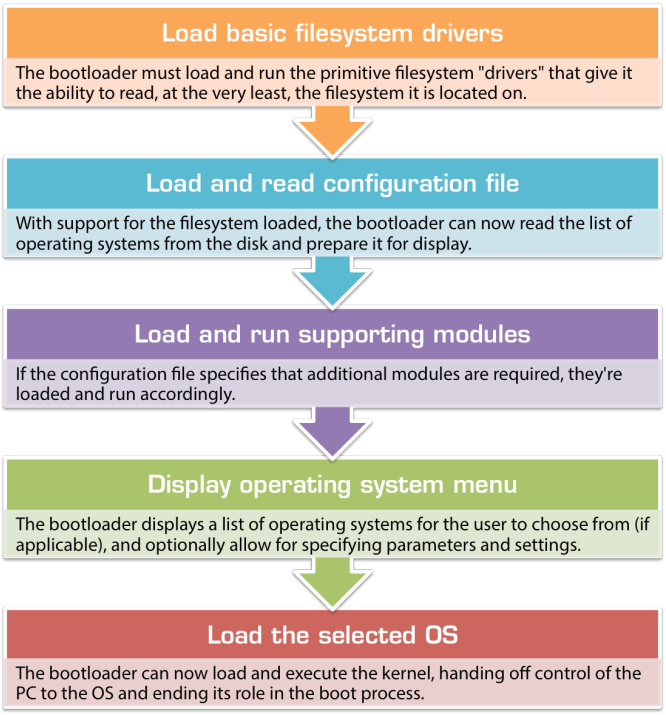
- First 512 bytes of active partition
  - First 3 bytes: JMP instruction; skip XX bytes to bootstrap code
- Filesystem header: info specific to file system in volume
- Bootstrap code: Second-stage boot loader
  - Ends with JMP to next sector in partition for actual loader
- Together, they look up a file stored on partition as a regular file and tell CPU to execute the file
  - Stored as a normal file on the file system

Bootloader

- Technically, 3rd-stage boot loader (after MBR, boot part)
  - Contains executable + config files
  - Most support dual boot or multi-boot



Bootloader Boot process

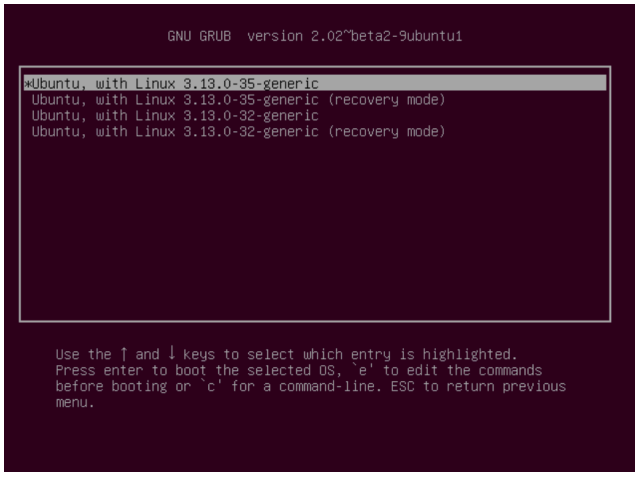


Sample /etc/grub2.cfg file

- menuentry defines kernels

```
terminal_output console
if [ x$feature_timeout_style = xy ] ; then
set timeout_style=menu
set timeout=5
# Fallback normal timeout code in case the timeout_style feature is
# unavailable.
else
set timeout=5
fi
### END /etc/grub.d/00_header ###
BEGIN /etc/grub.d/10_linux ###
menuentry 'CentOS Linux (3.10.0-123.4.2.el7.x86_64) 7 (Core)' --class centos --class gnu-l
inux --class gnu --class os --unrestricted $menuentry_id_option 'gnulinux-3.10.0-123.el7.x
86_64-advanced-fe0109f2-6f34-48ae-b51e-1f5fa78305b5' {
load_video
set gfxpayload=keep
insmod gzio
insmod part_msdos
insmod ext2
set root='hd0,msdos1'
```


OS Kernel Boot



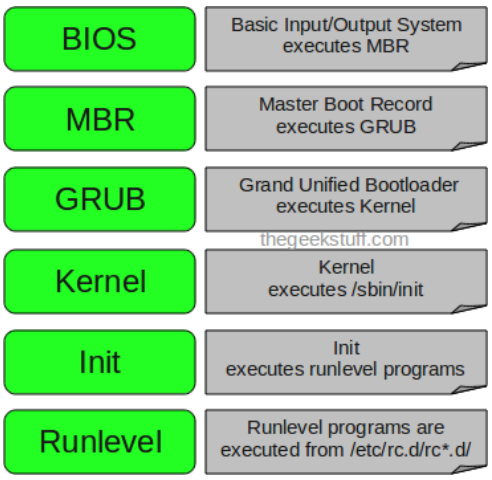


figure: courtesy thegeekstuff.com

OS Kernel Boot Steps (Linux)

- Select kernel from GRUB
  - Linux kernel is often stored in compressed form
- Load kernel
- Image file containing root FS and all kernel modules loaded (/boot/initramfs) [old name was initrd]
  - GRUB starts kernel, provides memory address of image file
  - Kernel mounts image file as initial memory-based root FS
  - Kernel detects hardware
  - Root fs on disk takes over from the memory-based root FS
- Boot process start INIT (SYSTEMD)

Boot process: INIT

- Kernel, once loaded, starts init (from /sbin/init)
  - on some systems init linked to "systemd"
  - init process becomes the first process in the system
    - all process are descendants of init
  - init process reads /etc/inittab as initial config file
  - Find selected **run level** and start services from /etc/rc directory
  - run level defines what services are started
    - 0: halt
    - 1: single user mode
    - 2 multi-user mode, no NSF
    - 3: full multi-user mode
    - 5: X11, 6: reboot [4 is unused]

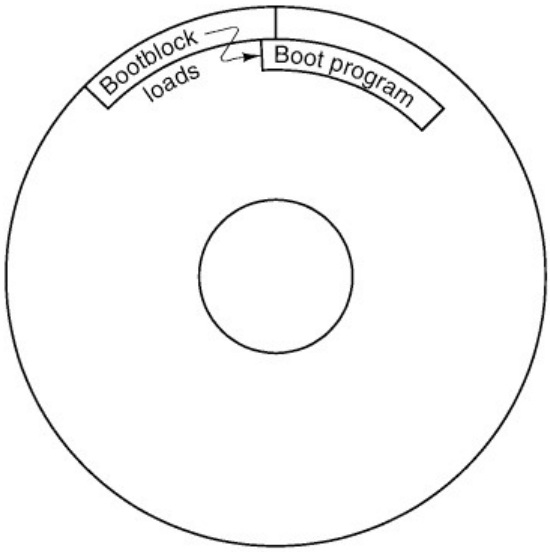
run-level Services

- Depending on init run-level, init will run all scripts from a certain directory to start services
  - To start a service at a run-level, create a startup script in appropriate directory
  - run level 0: /etc/rc.d/rc0.d/
  - run level 1: /etc/rc.d/rc1.d
  - run level 3: /etc/rc.d/rc3.d
- Scripts starting with "S" are startup; starting with "K" for kill during shutdown
- Filenames are started in sequence
  - S12syslog (seq # is 12), S80sendmail (seq # 80)…
- Key services: dhcp, networking, sshd started here

MINIX 3 Startup

Component	Description	Loaded by
kernel	Kernel + clock and system tasks	(in boot image)
pm	Process manager	(in boot image)
fs	File system	(in boot image)
rs	(Re)starts servers and drivers	(in boot image)
memory	RAM disk driver	(in boot image)
log	Buffers log output	(in boot image)
tty	Console and keyboard driver	(in boot image)
driver	Disk (at, bios, or floppy) driver	(in boot image)
init	parent of all user processes	(in boot image)
floppy	Floppy driver (if booted from hard disk)	/etc/rc
is	Information server (for debug dumps)	/etc/rc
cmos	Reads CMOS clock to set time	/etc/rc
random	Random number generator	/etc/rc
printer	Printer driver	/etc/rc

Bootstrapping MINIX (1)

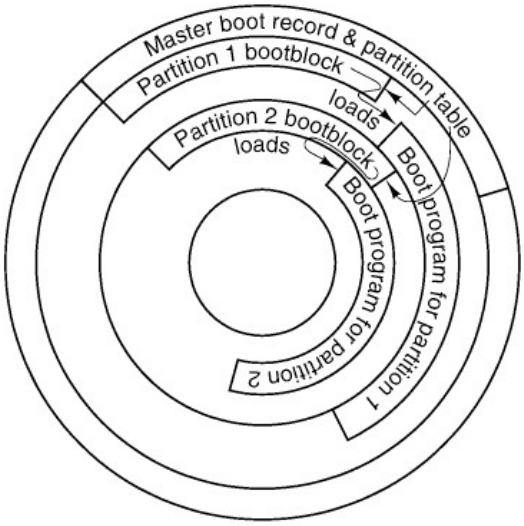


(a)

Disk structures used for bootstrapping.

Unpartitioned disk: The first sector is the bootblock.

Bootstrapping MINIX (2)



(b)

Disk structures used for bootstrapping.

Partitioned disk: The first sector is the master boot record, also called masterboot.

Boot Time in MINIX

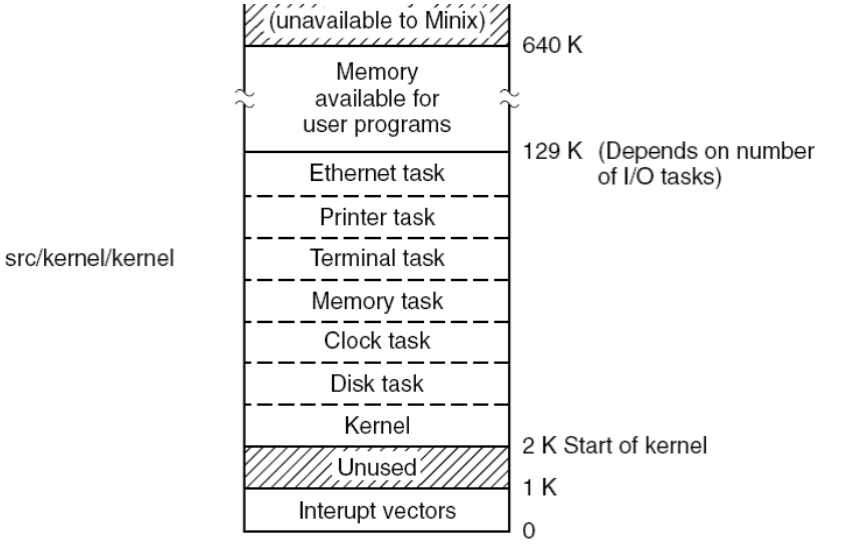
```
rootdev=256
ramimagedev=916
ramsize=4096
processor=586
bus=at
video=vga
chrome=color
memory=800:92880,100000:2F00000
c0=at
image=/minix/2.0.3r5
```

Boot parameters passed to the kernel at boot time in a typical MINIX 3 system.

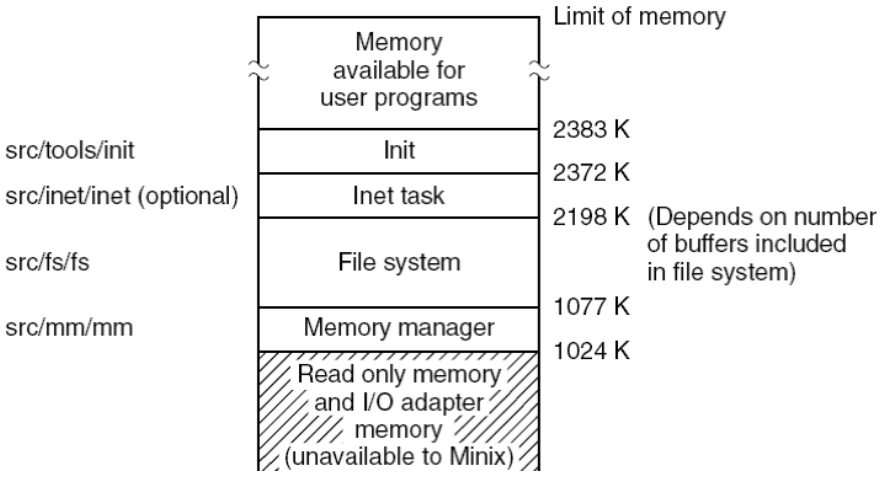
Minix boot

- Use /dev/imgrd as ram disk
- Unlike linux, no rc.d directories
- Start **/etc/rc** script defines what services to start
- **/etc/system.conf** defines rights for system services

MINIX Memory



MINIX Memory



Memory layout after MINIX 3 has been loaded from the disk into memory. The kernel, servers, and drivers are independently compiled and linked programs, listed on the left.