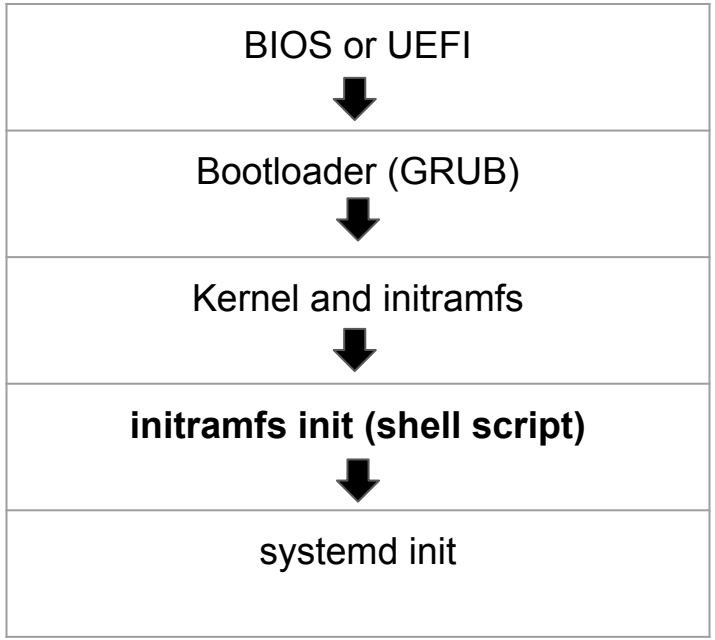
CS 447/647

Booting and System Management

Overview

- Finding, loading, and running bootstrapping code
- Finding, loading, and running the OS kernel
- Running startup scripts and system daemons
- Maintaining process hygiene and managing system state transitions

Virtual Machine Boot



Booting

- Power-on
- Power-on Self Test
- First Stage Bootloader
- Second Stage Bootloader
- Kernel starts

physical

- Kernel loads drivers and initializes hardware
- init starts
- system processes / daemons start
- DNS server starts and binds network socket
- DHCP server starts

userspace

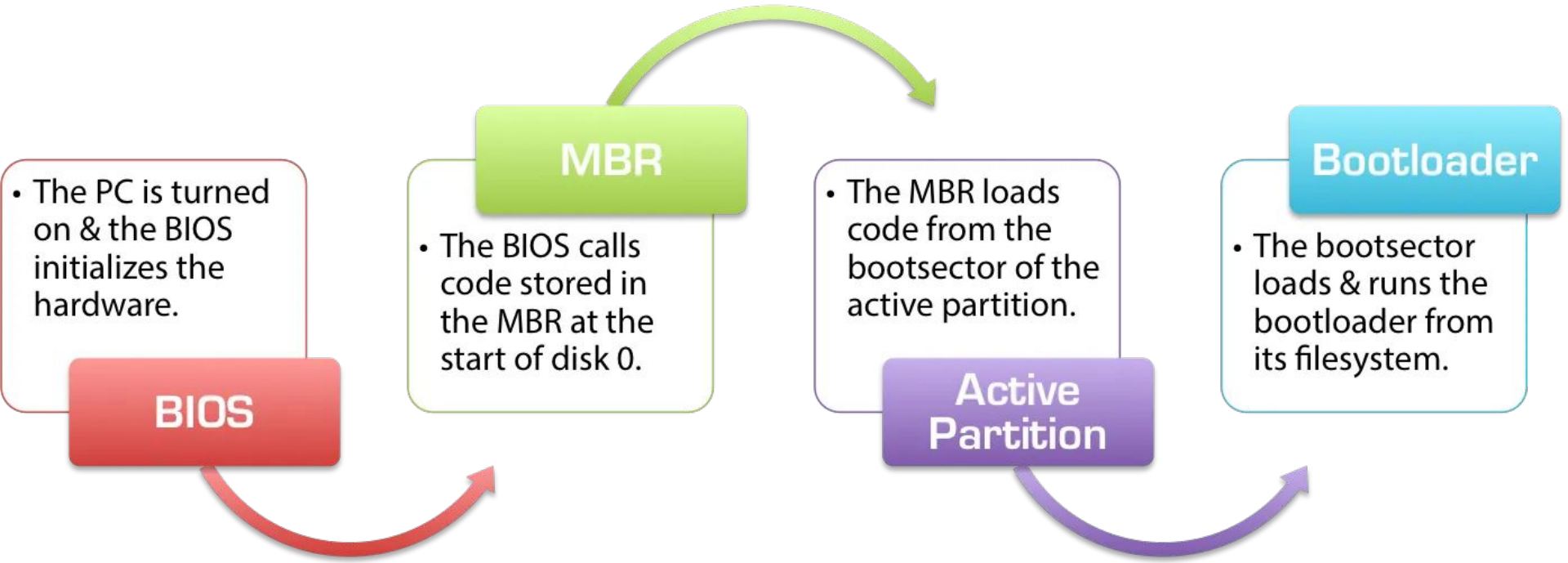
Early boot process

“Administrators have little direct, interactive control over most of the steps required to boot a system.”

- Basic Input Output System (BIOS) or
  Unified Extensible Firmware Interface (UEFI)
  - BIOS is legacy
  - UEFI is the current revision of EFI
- Power-On Self Test - P.O.S.T
  - Short Test of Hardware
  - Processors, RAM and Graphics (GPU)
  - Compatibility

BIOS - Basic Input Output System

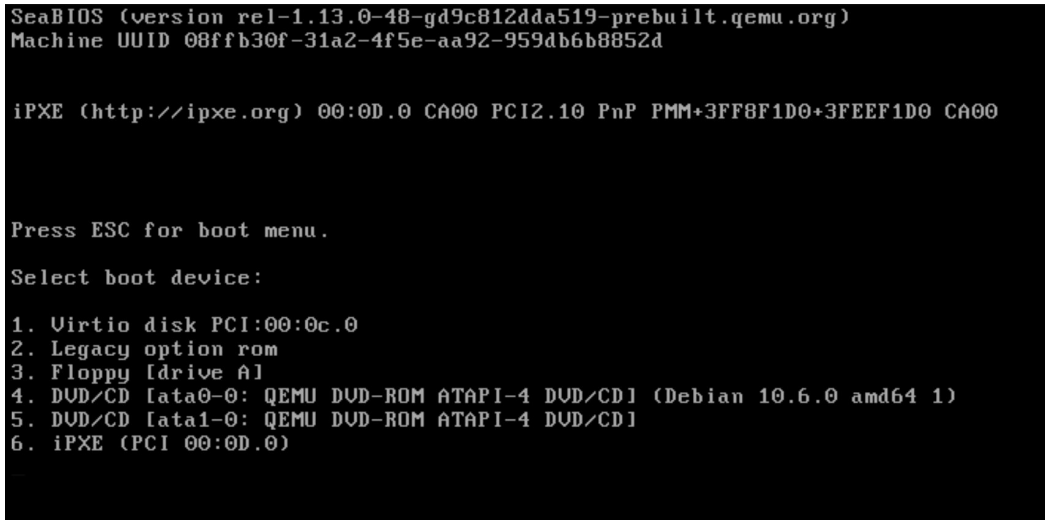
- 1st time hardware meets software
  - lowest level software
- Created in 1975
- Real Mode - 16-bit
  - Backwards Compatibility
- Provides
  - *USB Support*
  - Boot Priority
  - Boot Menu
  - PCI Configuration
  - CPU + RAM Configuration

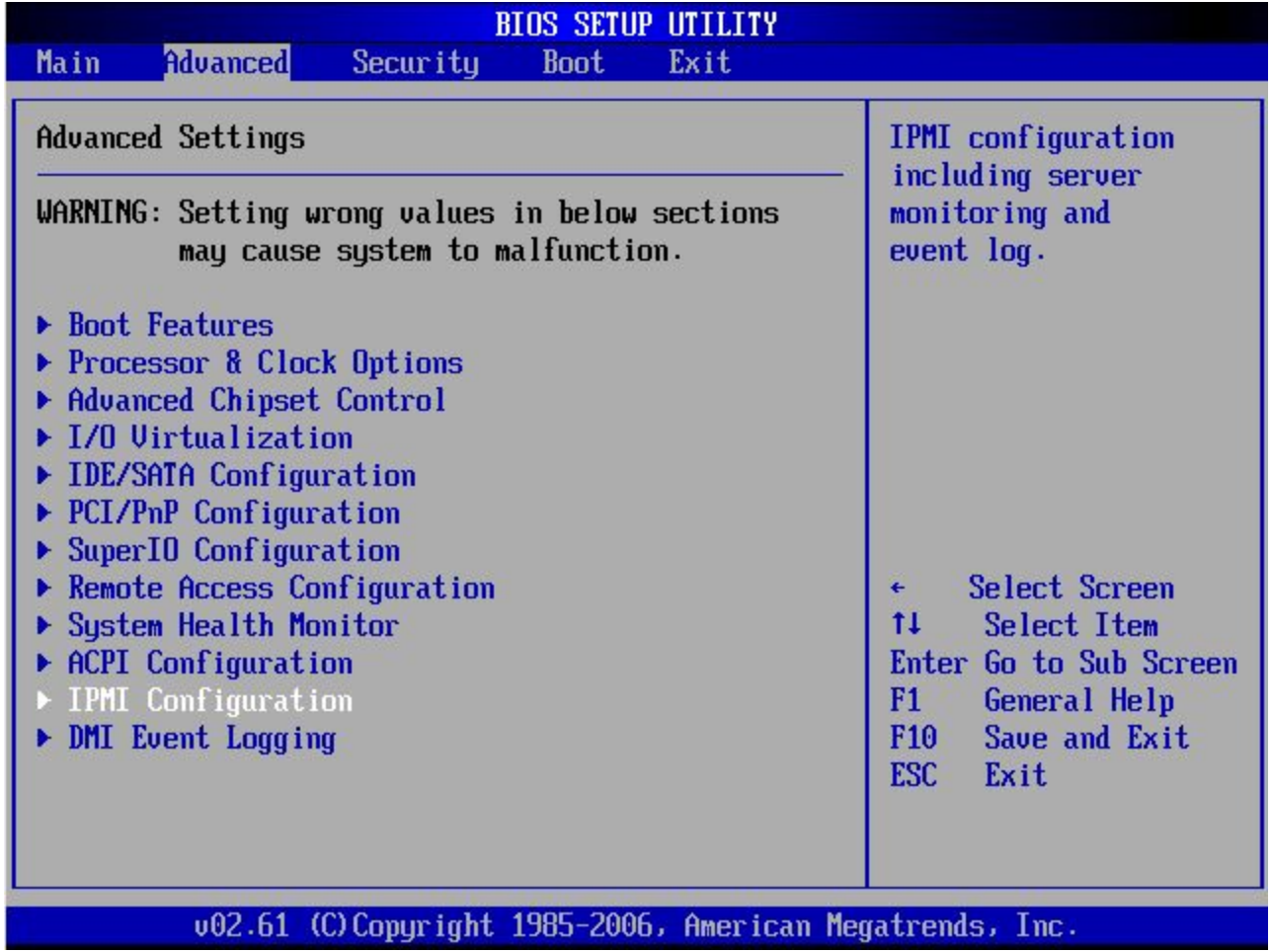


“The BIOS/MBR Boot Process.” NeoSmart Knowledgebase, 28 Feb. 2015, neosmart.net/wiki/mbr-boot-process/.

Why is BIOS important?

- It is still used today.
  - Legacy Mode
  - Virtual Machines - QEMU
    - https://github.com/coreboot/seabios




BIOS SETUP UTILITY
Main Advanced Security Boot Exit
Advanced Settings
WARNING: Setting wrong values in below sections
may cause system to malfunction.
Boot Features
Processor & Clock Options
Advanced Chipset Control
I/O Virtualization
IDE/SATA Configuration
PCI/PnP Configuration
SuperIO Configuration
Remote Access Configuration
System Health Monitor
ACPI Configuration
IPMI Configuration
DMI Event Logging
IPMI configuration including server monitoring and event log.
← Select Screen
↑↓ Select Item
Enter Go to Sub Screen
F1 General Help
F10 Save and Exit
ESC Exit
v02.61 (C)Copyright 1985-2006, American Megatrends, Inc.

BIOS+MBR

- Stage 1 - Boots with Master Boot Record (MBR)
  - Boot block - The first 512B (446B for bootstrapping) of the disk
- Stage 1.5- core.img
  - Drivers for the Filesystem
  - Before the 64th disk block. ~32Kb of storage
    - 1MiB Reserved for “stuff”
  - Partition selected by the “boot” flag
- Stage 2 - Execute the bootloader (GRUB).
  - Chainloading
- **Downsides**
  - Maximum disk size <= 2TiB
  - Hardware support.
  - 4 primary partitions

```
dd if=/dev/vda of=/tmp/mbr.bin bs=512 count=1
```

```
fdisk -l ./mbr.bin

Disk ./mbr.bin: 512 B, 512 bytes, 1 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xc8571a47

Device     Boot Start       End   Sectors Size Id Type
./mbr.bin1 *     2048 134217727 134215680  64G 83 Linux
```

UEFI - Unified Extensible Firmware Interface

- GUID Partition Table
  - Modern disk partitioning scheme
  - EFI System Partition (ESP) - FAT32 partition for grub, kernels and initramfs
- No bootloader is technically required
  - Most use a bootloader for legacy support
  - EFISTUB - https://wiki.archlinux.org/index.php/EFISTUB
- Provides a shell
  - Modify variables
  - Partitioning programs
  - Loading drivers
  - Edit files
- Intel, ARM, AMD, AMI, Apple, Dell, Microsoft, IBM, Lenovo, HP

`efibootmgr(8)`

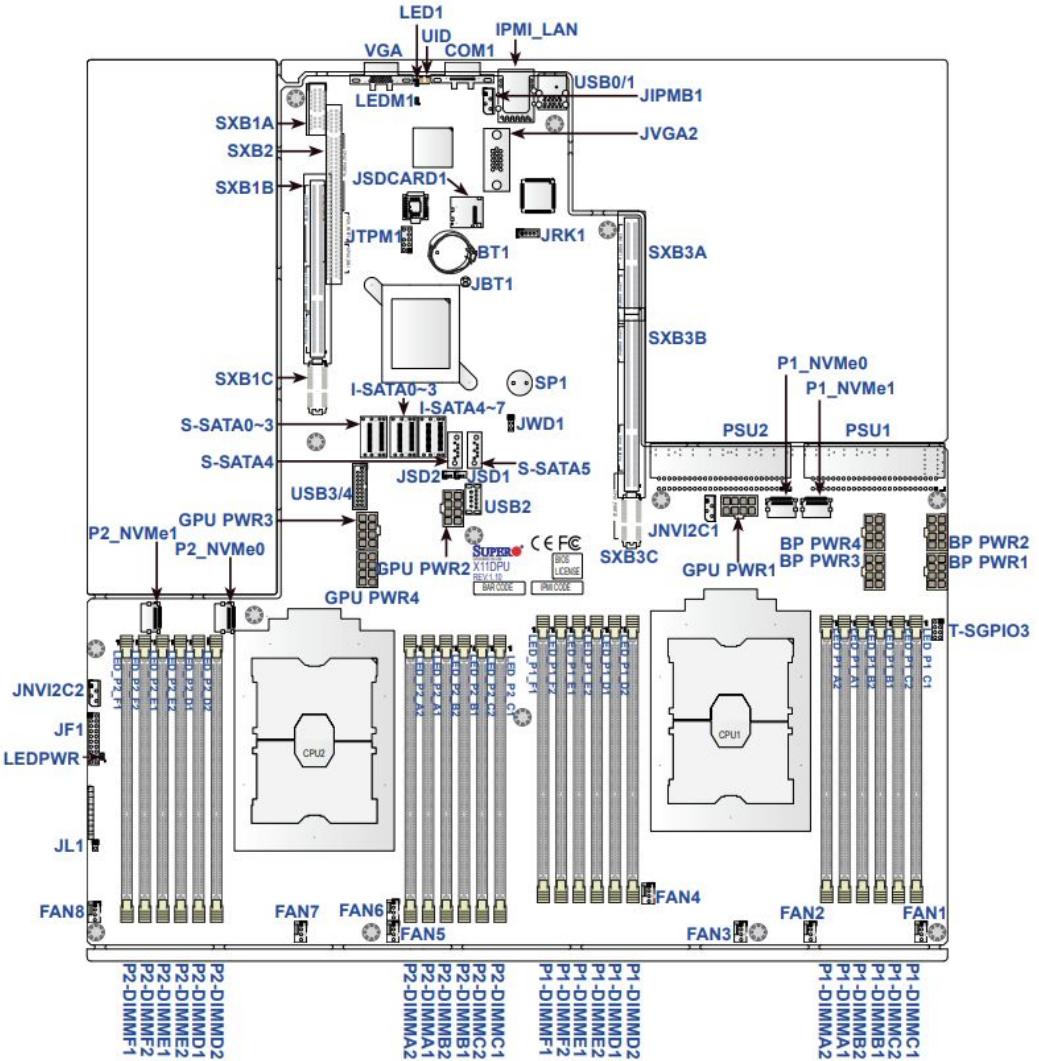
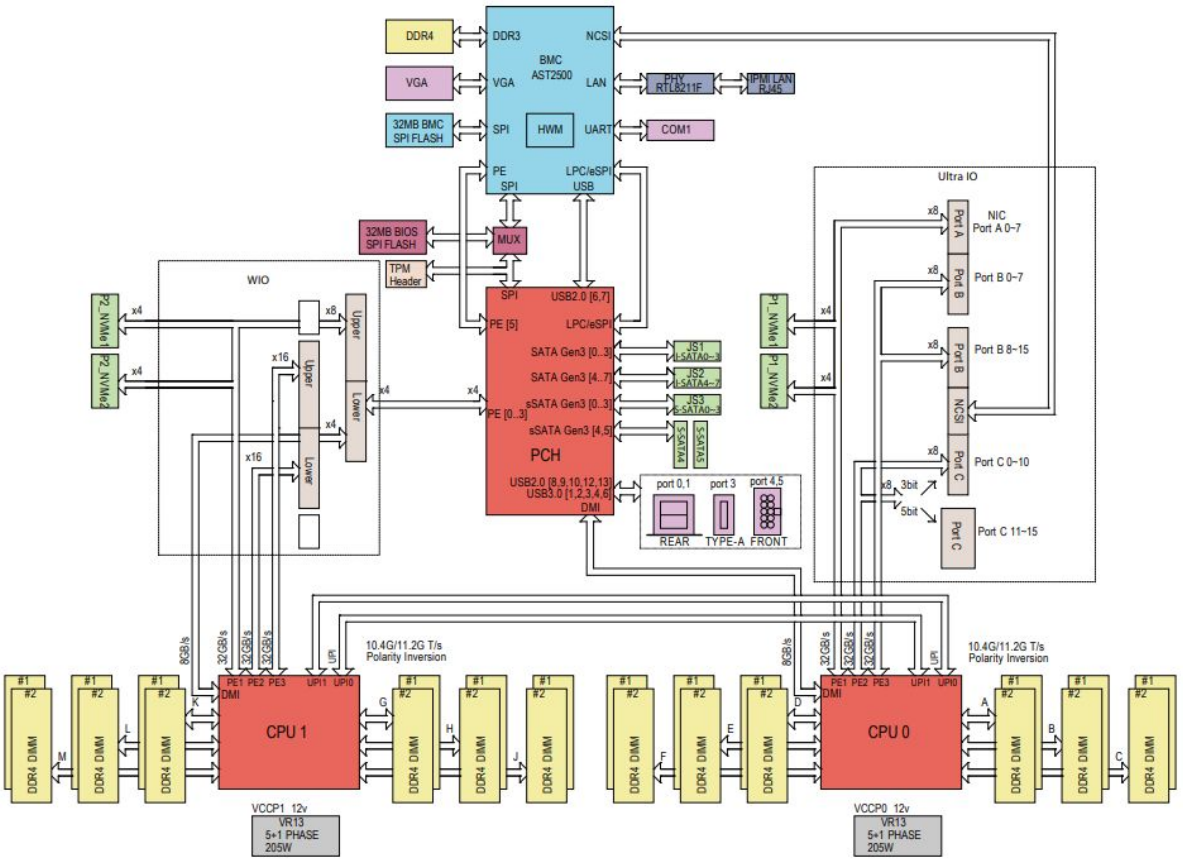
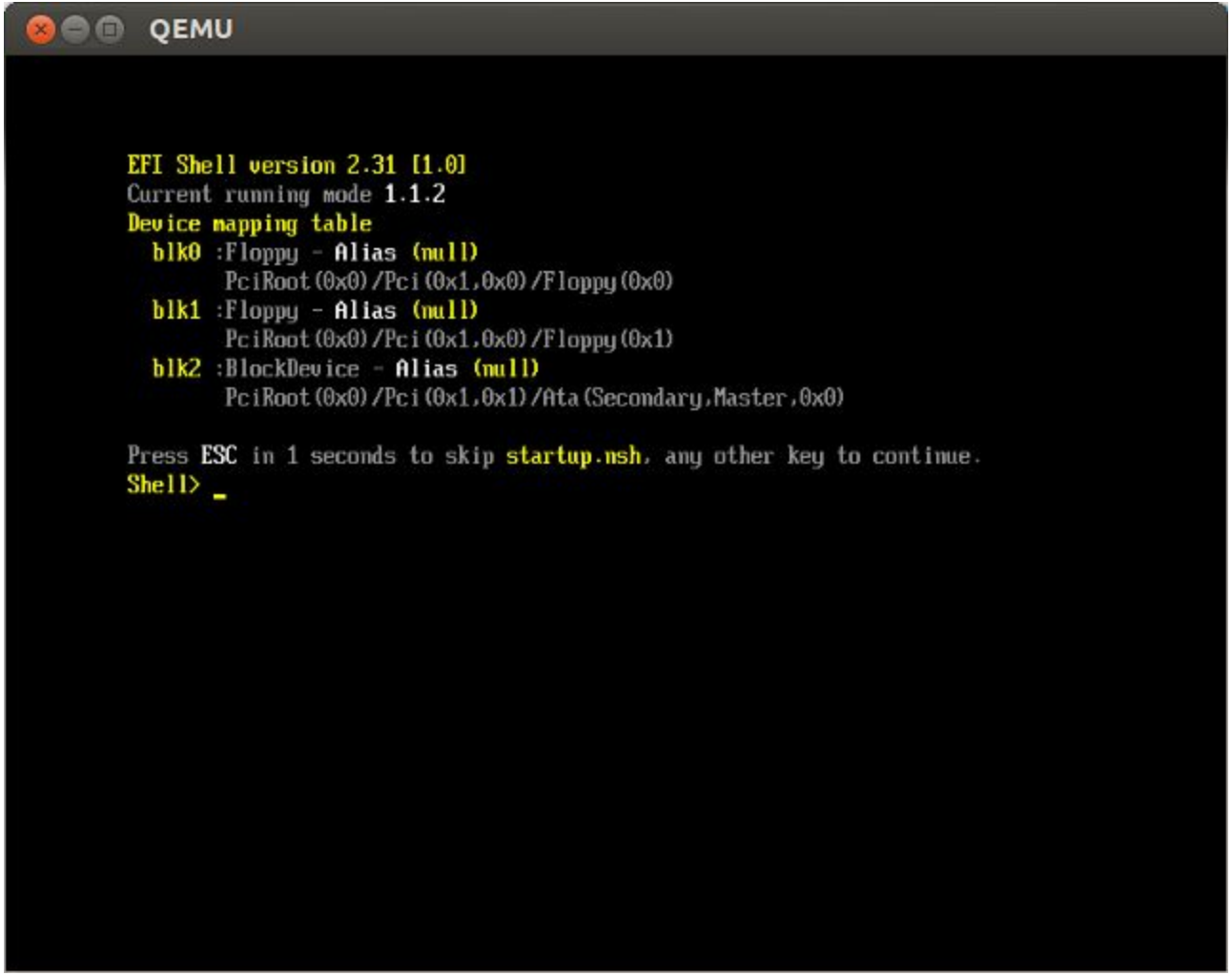

LED1
UID
VGA
COM1
IPMI_LAN
LEDM1
USB0/1
JIPMB1
JVGA2
SXB1A
SXB2
SXB1B
JSDCARD1
JTPM1
JRK1
BT1
JBT1
SXB3A
SXB3B
SXB1C
I-SATA0~3
SP1
P1_NVMe0
P1_NVMe1
I-SATA4~7
JWD1
PSU2
PSU1
S-SATA0~3
S-SATA4
JSD2
JSD1
S-SATA5
USB3/4
USB2
JNVI2C1
BP PWR4
BP PWR3
BP PWR2
BP PWR1
P2_NVMe1
GPU PWR3
P2_NVMe0
GPU PWR2
GPU PWR1
SXB3C
GPU PWR4
T-SGPIO3
JNVI2C2
JF1
LEDPWR
JL1
FAN8
FAN7
FAN6
FAN5
FAN4
FAN3
FAN2
FAN1
CPU2
CPU1
P2-DIMMF1
P2-DIMMF2
P2-DIMME1
P2-DIMME2
P2-DIMMD1
P2-DIMMD2
P2-DIMMA2
P2-DIMMA1
P2-DIMMB2
P2-DIMMB1
P2-DIMMC2
P2-DIMMC1
P1-DIMMF1
P1-DIMMF2
P1-DIMME1
P1-DIMME2
P1-DIMMD1
P1-DIMMD2
P1-DIMMA2
P1-DIMMA1
P1-DIMMB2
P1-DIMMB1
P1-DIMMC2
P1-DIMMC1

Figure 1-3.
System Block Diagram



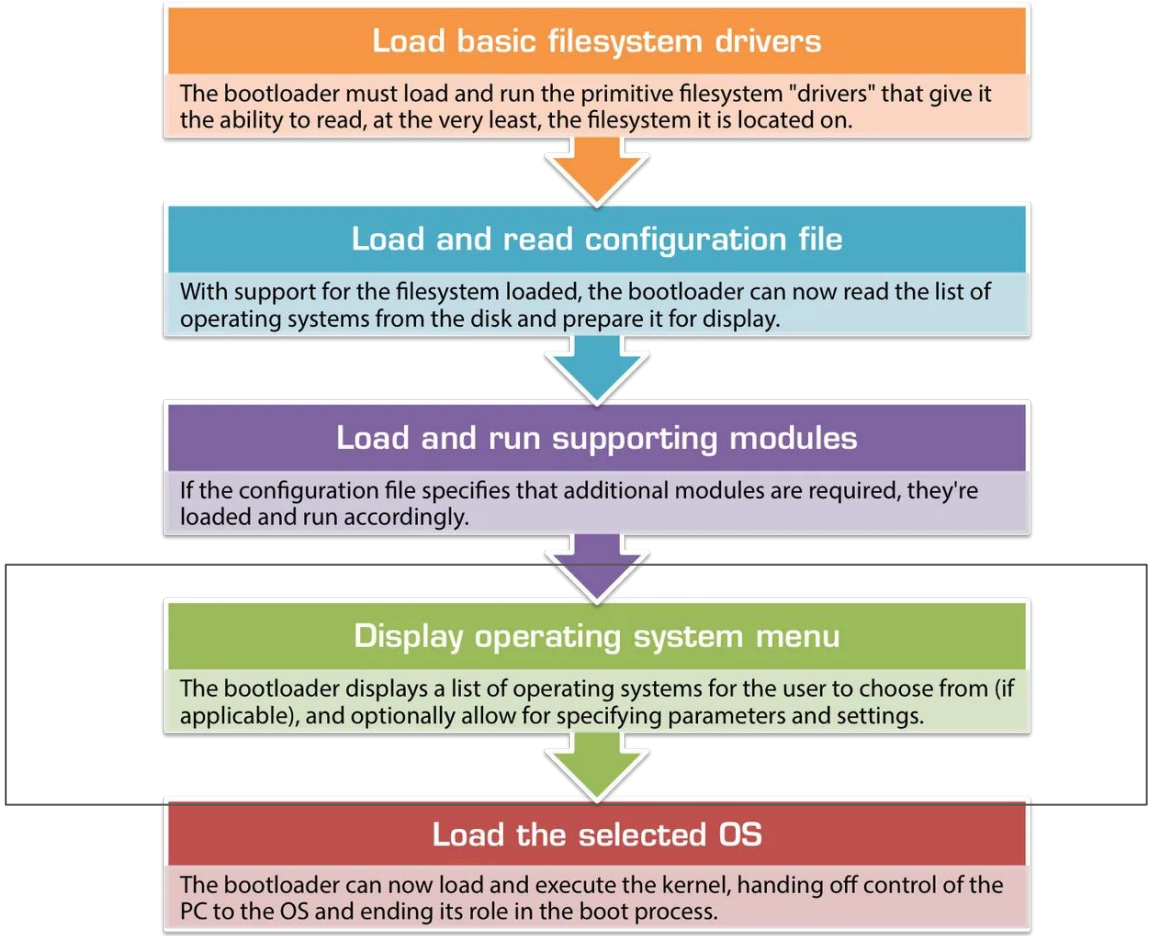
Virtual Machine

```
qemu-system-x86_64 -enable-kvm \
-name test \
-m 2048 \
-smp 4,sockets=1,cores=4,threads=1\
-vga std \
-usb \
-drive
if=pflash,format=raw,readonly,file=/usr/share/OVMF/OVMF_CODE.fd \
-drive if=pflash,format=raw,file=/var/local/OVMF_VARS_n.fd \
-drive format=raw,media=cdrom,readonly,file=ubuntu.iso \
-drive file=/var/local/n.img,if=virtio \
-cpu host,kvm=off
```


QEMU
EFI Shell version 2.31 [1.0]
Current running mode 1.1.2
Device mapping table
blk0 :Floppy - Alias (null)
PciRoot(0x0)/Pci(0x1,0x0)/Floppy(0x0)
blk1 :Floppy - Alias (null)
PciRoot(0x0)/Pci(0x1,0x0)/Floppy(0x1)
blk2 :BlockDevice - Alias (null)
PciRoot(0x0)/Pci(0x1,0x1)/Ata(Secondary,Master,0x0)
Press ESC in 1 seconds to skip startup.nsh, any other key to continue.
Shell>

GRUB- GRand Unified Boot loader

- Available in Ubuntu since 9.10 (October 2009)
  - LILO prior
- grub2 the default
  - grub-pc for BIOS
  - grub-efi for EFI
- `/boot/grub/grub.cfg` stores the menu
- Most distro's include scripts for generating a grub.cfg
  - grub-mkconfig - Generates a config to stdout
  - update-grub2 - iterates over partitions and kernels to create a menu
  - grub-install - installs the stage1 and stage 1.5 bootloader



“The BIOS/MBR Boot Process.” NeoSmart Knowledgebase, 28 Feb. 2015, neosmart.net/wiki/mbr-boot-process/.

```
menuentry "Linux" {
  # assuming that UUID is 763A-9CB6
  search --no-floppy --set=root --fs-uuid 763A-9CB6
  # search by label OTHER_LINUX (make sure that partition label is unambiguous)
  # search --no-floppy --set=root --label OTHER_LINUX

  linux /boot/vmlinuz #add other options here as required, for example: root=UUID=763A-9CB6
  initrd /boot/initrd.img #if the other kernel uses/needs one
}
```

```
root@zachnewell:~# lsblk -fs /dev/mapper/loop0p1
NAME    FSTYPE LABEL UUID                                 MOUNTPOINT
loop0p1 vfat         3F38-4569
└─loop0
```

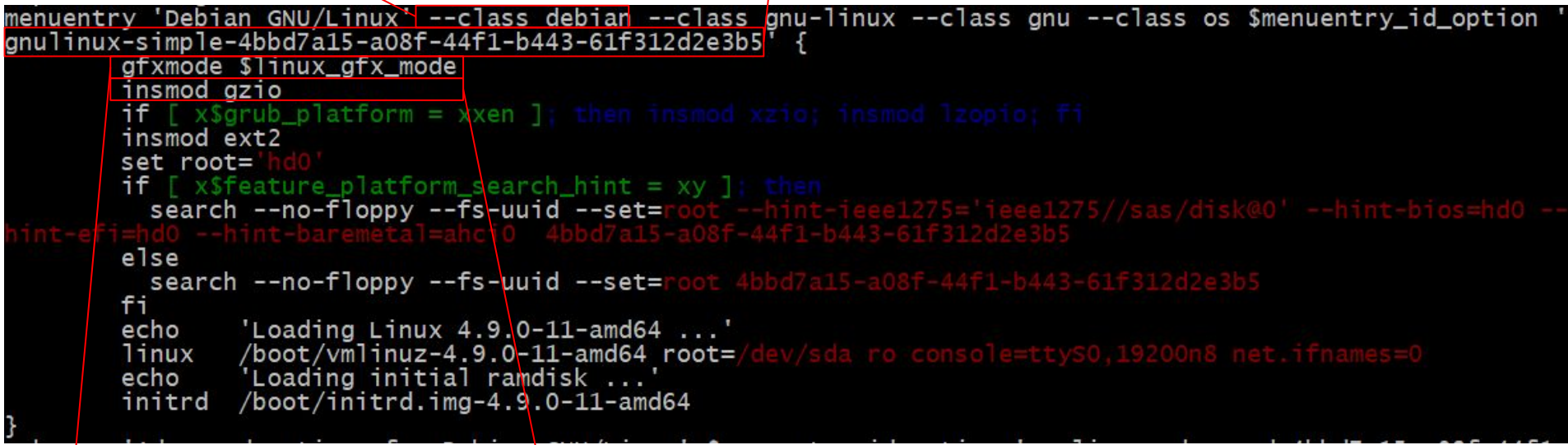
```
root@zachnewell:~# parted -s /dev/loop0 "print"
Model: Loopback device (loopback)
Disk /dev/loop0: 10.7GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start  End    Size   Type     File system  Flags
 1      512B   500MB  500MB  primary  fat16        boot, lba
```

https://wiki.archlinux.org/index.php/GRUB#Boot_menu_entry_examples

What a update-grub2 grub.cfg looks like...

Allows grouping of entries

--id may be used to associate unique identifier with a menu entry. GRUB_DEFAULT



Defaults to `auto` or `text`.

Dynamically inserts a grub module.

insmod

```
root@zachnewell:/boot/grub/i386-pc# ls -1 | wc -l
278
```

Important ones:

Windows

- Filesystem: fat.mod, ntfs.mod, ext2.mod, exfat.mod, xfs.mod, zfs.mod
- Block Device: mdraid1x.mod, lvm.mod
- Compression: lzopio.mod, gzio.mod, xzio.mod
- Important: serial.mod, luks.mod

```
menuentry 'Debian GNU/Linux' --class debian --class gnu-linux --class gnu --class os $menuentry_id_option '
gnulinux-simple-4bbd7a15-a08f-44f1-b443-61f312d2e3b5' {
        gfxmode $linux_gfx_mode
        insmod gzio
        if [ x$grub_platform = xxen ]; then insmod xzio; insmod lzopio; fi
        insmod ext2
        set root='hd0'
        if [ x$feature_platform_search_hint = xy ]; then
          search --no-floppy --fs-uuid --set=root --hint-ieee1275='ieee1275//sas/disk@0' --hint-bios=hd0 --
hint-efi=hd0 --hint-baremetal=ahci0  4bbd7a15-a08f-44f1-b443-61f312d2e3b5
        else
          search --no-floppy --fs-uuid --set=root 4bbd7a15-a08f-44f1-b443-61f312d2e3b5
        fi
        echo    'Loading Linux 4.9.0-11-amd64 ...'
        linux   /boot/vmlinuz-4.9.0-11-amd64 root=/dev/sda ro console=ttyS0,19200n8 net.ifnames=0
        echo    'Loading initial ramdisk ...'
        initrd  /boot/initrd.img-4.9.0-11-amd64
}
```

`linux kernel_filepath kernel_args`

Path to the initial RAM filesystem initramfs

GRUB - /etc/default/grub

Shell variable name	Contents or function
GRUB_BACKGROUND	Background image [a]
GRUB_CMDLINE_LINUX	Kernel parameters to add to menu entries for Linux [b]
GRUB_DEFAULT	Number or title of the default menu entry
GRUB_DISABLE_RECOVERY	Prevents the generation of recovery mode entries
GRUB_PRELOAD_MODULES	List of GRUB modules to be loaded as early as possible
GRUB_TIMEOUT	Seconds to display the boot menu before autoboot

a. The background image must be a **.png**, **.tga**, **.jpg**, or **.jpeg** file.
b. Table 2.3 lists some of the available options.

grub2 commands

Cmd	Function
boot	Boots the system from the specified kernel image
help	Gets interactive help for a command
linux	Loads a Linux kernel
reboot	Reboots the system
search	Searches devices by file, filesystem label, or UUID
usb	Tests USB support

Kernel

- Interface between hardware and software.
  - Drivers - SATA, SCSI, USB, PCIe, RAID
- Monolithic
  - Modular, lsmod, rmmod insmod, and modprobe
- Provides interfaces to hardware and low-level systems
  - System Calls
  - /sys/devices

```
root@cs447:/sys/bus/usb/devices/usb1# ls
1-0:1.0              bMaxPacketSize0     descriptors  interface_authorized_default  remove
authorized           bMaxPower           dev          ltm_capable                  serial
authorized_default   bNumConfigurations  devnum       manufacturer                 speed
avoid_reset_quirk    bNumInterfaces      devpath      maxchild                     subsystem
bConfigurationValue  bcdDevice           driver       power                        uevent
bDeviceClass         bmAttributes        ep_00        product                      urbnum
bDeviceProtocol      busnum              idProduct    quirks                       version
bDeviceSubClass      configuration       idVendor     removable
root@cs447:/sys/bus/usb/devices/usb1# ls power/
active_duration       level                     runtime_usage        wakeup_expire_count
async                 runtime_active_kids       wakeup               wakeup_last_time_ms
autosuspend           runtime_active_time       wakeup_abort_count   wakeup_max_time_ms
autosuspend_delay_ms  runtime_enabled           wakeup_active        wakeup_total_time_ms
connected_duration    runtime_status            wakeup_active_count
control               runtime_suspended_time    wakeup_count
```

```
# If you change this file, run 'update-grub' afterwards to update
# /boot/grub/grub.cfg.
# For full documentation of the options in this file, see:
#   info -f grub -n 'Simple configuration'

GRUB_DEFAULT=0
GRUB_TIMEOUT=5
GRUB_DISTRIBUTOR=`lsb_release -i -s 2> /dev/null || echo Debian`
GRUB_CMDLINE_LINUX_DEFAULT="quiet intel_iommu=on kvm-intel.nested=1"
GRUB_CMDLINE_LINUX=""

# Uncomment to enable BadRAM filtering, modify to suit your needs
# This works with Linux (no patch required) and with any kernel that obtains
# the memory map information from GRUB (GNU Mach, kernel of FreeBSD ...)
#GRUB_BADRAM="0x01234567,0xfefefefe,0x89abcdef,0xefefefef"

# Uncomment to disable graphical terminal (grub-pc only)
#GRUB_TERMINAL=console

# The resolution used on graphical terminal
# note that you can use only modes which your graphic card supports via VBE
# you can see them in real GRUB with the command `vbeinfo'
#GRUB_GFXMODE=640x480

# Uncomment if you don't want GRUB to pass "root=UUID=xxx" parameter to Linux
#GRUB_DISABLE_LINUX_UUID=true

# Uncomment to disable generation of recovery mode menu entries
#GRUB_DISABLE_RECOVERY="true"

# Uncomment to get a beep at grub start
#GRUB_INIT_TUNE="480 440 1"
```

#Disable most log messages
GRUB_CMDLINE_LINUX_DEFAULT="quiet"

#Disable power conservation
GRUB_CMDLINE_LINUX_DEFAULT="intel_idle.max_cstate=0 processor.max_cstate=1 intel_pstate=disable"

#Change init
GRUB_CMDLINE_LINUX_DEFAULT="init=/bin/bash"

U-Boot

- Open-Source Stage 2 Bootloader
- Primarily for embedded Linux
  - ARM
- Uses a UART or Serial Port for output
- Supports
  - DHCP - Dynamic Host Control Protocol
  - TFTP -Trivial File Transfer Protocol
  - GPIO Manipulation - General Purpose Input Output
  - MMC - Block device
  - Networking - UDP, ICMP, ARP
  - Loading the kernel over serial via modem commands

initramfs

- The initramfs is a gzipped ***cpio*** archive.
- At boot time, the kernel unpacks that archive into a RAM disk,
- It mounts and uses it as initial root file system.
- The finding of the root device happens in this early userspace.
- Generated with `update-initramfs`
  - `/etc/initramfs-tools/update-initramfs.conf`

```
man update-initramfs 5

man update-initramfs 8
```

Why cpio?

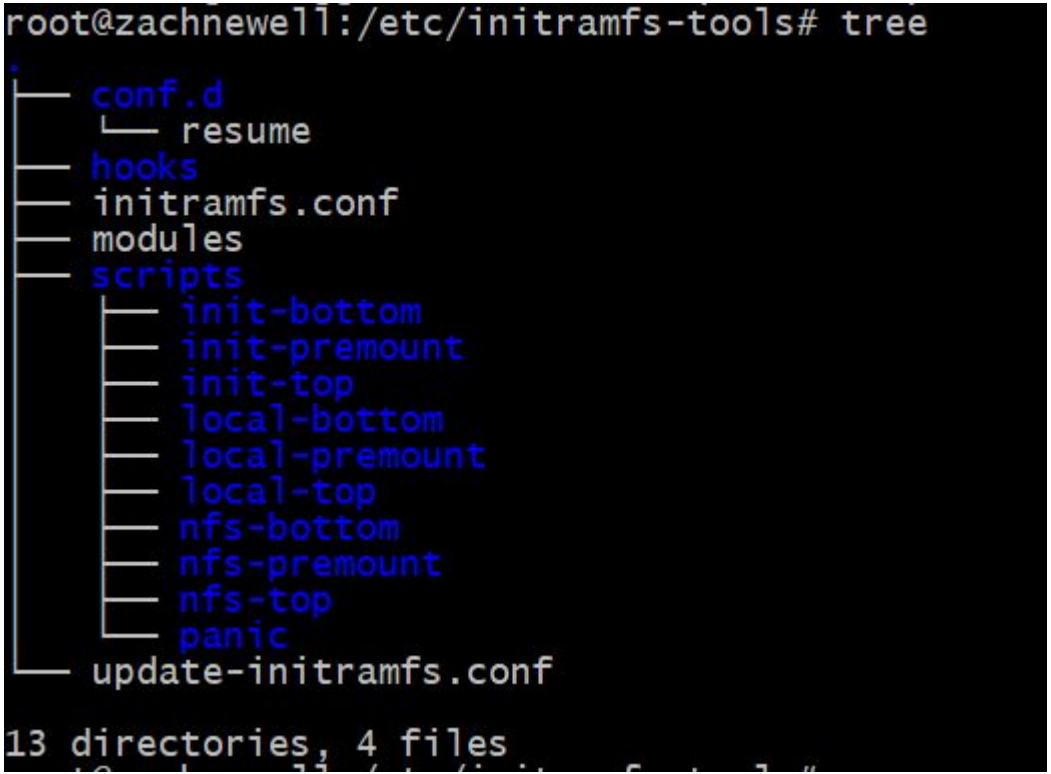
1. It’s a standard format. Device Drivers. 1996
    a. Not as popular as tar because the cmdline arguments are horrendous.
2. Simpler and cleaner
    a. Spec is 26k of text
3. tar hasn’t been standardized.
4. Kernel internal format. Already existed inside the kernel.
5. Al Viro (kernel developer) made the decision
    a. "tar is ugly as hell and not going to be supported on the kernel side"

https://www.kernel.org/doc/Documentation/filesystems/ramfs-rootfs-initramfs.txt

initramfs

```
root@zachnewell:/etc/initramfs-tools# grep -v "^#" initramfs.conf

MODULES=most

BUSYBOX=auto

KEYMAP=n

COMPRESS=gzip


DEVICE=

NFSROOT=auto
```

initramfs


root@zachnewell:/etc/initramfs-tools# tree
.
├── conf.d
│   └── resume
├── hooks
├── initramfs.conf
├── modules
├── scripts
│   ├── init-bottom
│   ├── init-premount
│   ├── init-top
│   ├── local-bottom
│   ├── local-premount
│   ├── local-top
│   ├── nfs-bottom
│   ├── nfs-premount
│   ├── nfs-top
│   └── panic
└── update-initramfs.conf
13 directories, 4 files

busybox

Swiss Army Knife of Embedded Linux

- Combines tiny versions of many common UNIX utilities
  - ls, bash(ash), cat, chown, chmod, mv, uniq, less, mount, umount
- Multi-call binary
  - /bin/busybox ls
  - Symlinked to /bin, IE: ln -s /bin/busybox /bin/ls
- Can be compiled with a different number of functions
  - Ubuntu by default has a lot less in initramfs-tools.
- Why?
  - Small

busybox - all-in-one

```
[, [[, acpid, adjtimex, ar, arch, arp, arping, ash, awk, basename, bc,
blkdiscard, blockdev, brctl, bunzip2, bzcat, bzip2, cal, cat, chgrp,
chmod, chown, chroot, chvt, clear, cmp, cp, cpio, cttyhack, cut, date,
dc, dd, deallocvt, depmod, devmem, df, diff, dirname, dmesg,
dnsdomainname, dos2unix, du, dumpkmap, dumpleases, echo, egrep, env,
expand, expr, factor, fallocate, false, fatattr, fgrep, find, fold,
free, freeramdisk, fsfreeze, fstrim, ftpget, ftpput, getopt, getty,
grep, groups, gunzip, gzip, halt, head, hexdump, hostid, hostname,
httpd, hwclock, i2cdetect, i2cdump, i2cget, i2cset, id, ifconfig,
ifdown, ifup, init, insmod, ionice, ip, ipcalc, ipneigh, kill, killall,
klogd, last, less, link, linux32, linux64, linuxrc, ln, loadfont,
loadkmap, logger, login, logname, logread, losetup, ls, lsmod, lsscsi,
lzcat, lzma, lzop, md5sum, mdev, microcom, mkdir, mkdosfs, mke2fs,
mkfifo, mknod, mkpasswd, mkswap, mktemp, modinfo, modprobe, more,
mount, mt, mv, nameif, nc, netstat, nl, nologin, nproc, nsenter,
nslookup, nuke, od, openvt, partprobe, paste, patch, pidof, ping,
ping6, pivot_root, poweroff, printf, ps, pwd, rdate, readlink,
realpath, reboot, renice, reset, resume, rev, rm, rmdir, rmmod, route,
rpm, rpm2cpio, run-init, run-parts, sed, seq, setkeycodes, setpriv,
setsid, sh, sha1sum, sha256sum, sha512sum, shred, shuf, sleep, sort,
ssl_client, start-stop-daemon, stat, strings, stty, svc, svok, swapoff,
swapon, switch_root, sync, sysctl, syslogd, tac, tail, tar, taskset,
tee, telnet, test, tftp, time, timeout, top, touch, tr, traceroute,
traceroute6, true, truncate, tty, ubirename, udhcpc, udhcpd, uevent,
umount, uname, uncompress, unexpand, uniq, unix2dos, unlink, unlzma,
unshare, unxz, unzip, uptime, usleep, uudecode, uuencode, vconfig, vi,
w, watch, watchdog, wc, wget, which, who, whoami, xargs, xxd, xz,
xzcat, yes, zcat
```