

# Booting Unix

- ☞ Steps in the typical Unix boot process
- ☞ The initial bootstrap program resides in firmware somewhere (e.g., the Sun monitor, PC BIOS mode)
  - ☞→ “man eeprom”, “man monitor” on Suns



# Booting Unix

⇒ “man grub” on Linux (grub is covered in more detail below)



# UNIX/Linux Kernel initialization

1. You can see running monologue from various kernel routines, mostly dealing with the kernel and device drivers recognizing the hardware of the system. (You may have to use CTRL-ALT-F1 / CTRL-ALT-F2 / etc. to find the virtual terminal output. Also, if you are coming up with X enabled, you may have to click on a button labeled something like “details” to see more.)



# UNIX/Linux Kernel initialization

2. Often logged in `/var/adm/messages` (or `/var/log/messages` or perhaps `/var/log/boot.log`) and typically the system console and usually accessible via **dmesg** (**dmesg** is not available on all UNIXes, alas)



# UNIX/Linux Kernel initialization

3. Tend to be very system-specific but with experience you should be able to scan a series of boot messages and spot problems.



# UNIX/Linux Kernel initialization

👉 Example **dmesg** outputs:



# UNIX/Linux Kernel initialization (old-ish 2.4)

```
Linux version 2.4.21-37.EL (centos@sillage.bis.pasteur.fr)
  (gcc version 3.2.3 20030502 (Red Hat Linux 3.2.3-52)) #1
  Wed Sep 28 14:14:23 EDT 2005
BIOS-provided physical RAM map:
  BIOS-e820: 0000000000000000 - 00000000000a0000 (usable)
  BIOS-e820: 00000000000f0000 - 0000000000100000 (reserved)
  BIOS-e820: 0000000000100000 - 000000001f771000 (usable)
  BIOS-e820: 000000001f771000 - 000000001f773000 (ACPI NVS)
```



# UNIX/Linux Kernel initialization (old-ish 2.4)

```
BIOS-e820: 000000001f773000 - 000000001f794000 (ACPI data)
BIOS-e820: 000000001f794000 - 000000001f800000 (reserved)
BIOS-e820: 00000000fec00000 - 00000000fec10000 (reserved)
BIOS-e820: 00000000fee00000 - 00000000fee10000 (reserved)
BIOS-e820: 00000000ffb00000 - 0000000100000000 (reserved)
```

OMB HIGHMEM available.

503MB LOWMEM available.

[ ... ]

Kernel command line: ro root=LABEL=/  
Initializing CPU#0





# UNIX/Linux Kernel initialization (old-ish 2.4)

```
Detected 2399.594 MHz processor.
```

```
Console: colour VGA+ 80x25
```

```
Calibrating delay loop... 4784.12 BogoMIPS
```

```
[ ... ]
```

```
Memory: 500836k/515524k available (1545k kernel code,  
12128k reserved, 1073k data, 164k init, 0k highmem)
```

```
[ ... ]
```

```
CPU: Trace cache: 12K uops, L1 D cache: 8K
```

```
CPU: L2 cache: 512K
```



# UNIX/Linux Kernel initialization (old-ish 2.4)

```
Intel machine check architecture supported.  
Intel machine check reporting enabled on CPU#0.  
CPU:      After generic, caps: bfebfbff 00000000 00000000 00000000  
CPU:      Common caps: bfebfbff 00000000 00000000 00000000  
CPU: Intel(R) Pentium(R) 4 CPU 2.40GHz stepping 07  
[ ... ]  
Linux NET4.0 for Linux 2.4  
[ ... ]
```



# UNIX/Linux Kernel initialization (old-ish 2.4)

```
hda: WDC WD800BB-75CAA0, ATA DISK drive
blk: queue c041c900, I/O limit 4095Mb (mask 0xffffffff)
hdc: LG CD-ROM CRN-8245B, ATAPI CD/DVD-ROM drive
ide0 at 0x1f0-0x1f7,0x3f6 on irq 14
ide1 at 0x170-0x177,0x376 on irq 15
hda: attached ide-disk driver.
hda: host protected area => 1
hda: setmax LBA 156301488, native 156250000
```



# UNIX/Linux Kernel initialization (old-ish 2.4)

```
hda: 156250000 sectors (80000 MB) w/2048KiB Cache,  
    CHS=9726/255/63, UDMA(100)  
ide-floppy driver 0.99.newide  
Partition check:  
    hda: hda1 hda2 hda3  
ide-floppy driver 0.99.newide
```



# UNIX/Linux Kernel initialization (old-ish 2.4)

```
[ ... ]
```

```
EXT3-fs: mounted filesystem with ordered data mode.
```

```
Freeing unused kernel memory: 164k freed
```

```
[ ... ]
```

```
EXT3 FS 2.4-0.9.19, 19 August 2002 on ide0(3,2), internal journal
```

```
Adding Swap: 1044216k swap-space (priority -1)
```



# UNIX/Linux Kernel initialization (old-ish 2.4)

```
kjournald starting.  Commit interval 5 seconds  
EXT3 FS 2.4-0.9.19, 19 August 2002 on ide0(3,1), internal journal  
EXT3-fs: mounted filesystem with ordered data mode.  
[ ... ]
```



# UNIX/Linux Kernel initialization (2.6)

```
Initializing cgroup subsys cpuset
```

```
Linux version 2.6.24.5-85.fc8 (mockbuild@xenbuilder2.fedora.redhat.com) (gcc version
```

```
Command line: ro root=/dev/VolGroup00/LogVol100 rhgb quiet
```

```
BIOS-provided physical RAM map:
```

```
  BIOS-e820: 0000000000000000 - 000000000000095000 (usable)
```

```
  BIOS-e820: 000000000000095000 - 000000000000a0000 (reserved)
```



# UNIX/Linux Kernel initialization (2.6)

```
ACPI: RSDP 000F99E0, 0014 (r0 ACPIAM)
ACPI: RSDT BFFA0000, 0048 (r1 ACRSYS ACRPRDCT 20070721 MSFT 97)
ACPI: FACP BFFA0200, 0084 (r2 072107 FACP1408 20070721 MSFT 97)
ACPI: DSDT BFFA05C0, 5BCE (r1 1AAAA 1AAAA000 0 INTL 20051117)
ACPI: FACS BFFAE000, 0040
ACPI: APIC BFFA0390, 006C (r1 072107 APIC1408 20070721 MSFT 97)
```





# UNIX/Linux Kernel initialization (2.6)

```
ACPI: PM-Timer IO Port: 0x808
ACPI: Local APIC address 0xfee00000
ACPI: LAPIC (acpi_id[0x01] lapic_id[0x00] enabled)
Processor #0 (Bootup-CPU)
ACPI: LAPIC (acpi_id[0x02] lapic_id[0x01] enabled)
Processor #1
ACPI: LAPIC (acpi_id[0x03] lapic_id[0x02] enabled)
Processor #2
ACPI: LAPIC (acpi_id[0x04] lapic_id[0x03] enabled)
Processor #3
```



# UNIX/Linux Kernel initialization (2.6)

```
SMP: Allowing 4 CPUs, 0 hotplug CPUs
PERCPU: Allocating 42248 bytes of per cpu data
Built 1 zonelists in Node order, mobility grouping on.  Total pages: 774141
Policy zone: DMA32
Kernel command line: ro root=/dev/VolGroup00/LogVol100 rhgb quiet
Initializing CPU#0
PID hash table entries: 4096 (order: 12, 32768 bytes)
hpet clockevent registered
TSC calibrated against HPET
time.c: Detected 2393.996 MHz processor.
Console: colour VGA+ 80x25
console [tty0] enabled
Checking aperture...
```



# UNIX/Linux Kernel initialization (2.6)

```
Memory: 3092116k/3145344k available (2491k kernel code, 52800k reserved, 1390k data
SLUB: Genslabs=12, HWalign=64, Order=0-1, MinObjects=4, CPUs=4, Nodes=1
Calibrating delay using timer specific routine.. 4790.59 BogoMIPS (lpj=2395297)
Security Framework initialized
SELinux:  Initializing.
SELinux:  Starting in permissive mode
selinux_register_security:  Registering secondary module capability
```



# UNIX/Linux Kernel initialization (2.6)

```
Booting processor 1/4 APIC 0x1
Initializing CPU#1
Calibrating delay using timer specific routine.. 4787.75 BogoMIPS (lpj=2393876)
CPU: L1 I cache: 32K, L1 D cache: 32K
CPU: L2 cache: 4096K
CPU 1/1 -> Node 0
CPU: Physical Processor ID: 0
CPU: Processor Core ID: 1
Intel(R) Core(TM)2 Quad CPU    Q6600  @ 2.40GHz stepping 0b
checking TSC synchronization [CPU#0 -> CPU#1]: passed.
```



# UNIX/Linux Kernel initialization (2.6)

```
ata6: SATA link down (SStatus 0 SControl 300)
scsi 0:0:0:0: Direct-Access      ATA          Hitachi HDT72505 V560 PQ: 0 ANSI: 5
sd 0:0:0:0: [sda] 976773168 512-byte hardware sectors (500108 MB)
sd 0:0:0:0: [sda] Write Protect is off
sd 0:0:0:0: [sda] Mode Sense: 00 3a 00 00
sd 0:0:0:0: [sda] Write cache: enabled, read cache: enabled, doesn't support DPO or
sd 0:0:0:0: [sda] 976773168 512-byte hardware sectors (500108 MB)
sd 0:0:0:0: [sda] Write Protect is off
sd 0:0:0:0: [sda] Mode Sense: 00 3a 00 00
sd 0:0:0:0: [sda] Write cache: enabled, read cache: enabled, doesn't support DPO or
sda: sda1 sda2 sda3 sda4 < sda5 sda6 >
sd 0:0:0:0: [sda] Attached SCSI disk
```



# UNIX/Linux `init` and `bootup` scripts

- 👉 UNIX/Linux operating systems use some series of programs and/or scripting that are started at boot time. Currently, most of these scripts are plain shell script text files. (However, this is an area of active research, and better dependency handling regimes are in the works.)
- 👉 Old-style BSD-based systems used a simple naming convention of `/etc/rc*`. This simplistic version of



startup scripts is rare these days. More common are the System V startup scripts (with many variations). See page 24 in USAH for system-specific details.



# UNIX/Linux bootup scripts

- ☞ SystemV bootup scripts allow for more complex script configurations (the scripts are “buried” in directories of `/etc/` rather than just being all in one directory). The `init` process starts the `rc` file processing.
- ⇒ `/etc/inittab` is the master config file for **init**





# UNIX/Linux bootup scripts

- This file controls which bootup scripts will be executed
- The scripts are divided into “run-levels”, which determine what sort of booting you are doing (multi-user, single-user, shutdown, specialty boot, etc.). The “man” page for “init” (“man init”) explains the run level numbering.



# UNIX/Linux bootup scripts

⇒ The idea is that soft-links that start with a capital “S” are executed at startup; soft-links with a leading capital “K” are executed at shutdown (you can see this behavior in `/sbin/rc2` on Solaris and `/etc/rc.d/rc` on RedHat).



# UNIX/Linux bootup scripts

- The Linux `init` package includes a nifty utility named **runlevel** that you can run to determine the current machine's run level.



# UNIX/Linux init

☞ Sample `/etc/inittab` file:

⇒ CentOS 3.6 Linux `/etc/inittab`



# UNIX Bootup scripts

```
#
# inittab          This file describes how the INIT process
#                 should set up
#                 the system in a certain run-level.
#
# Author:         Miquel van Smoorenburg,
#                 <miquels@drinkel.nl.mugnet.org>
#                 Modified for RHS Linux by Marc Ewing and
#                 Donnie Barnes
#
```



# UNIX init

```
# Default runlevel. The runlevels used by RHS are:
# 0 - halt (Do NOT set initdefault to this)
# 1 - Single user mode
# 2 - Multiuser, without NFS (The same as 3, if you do not have networking)
# 3 - Full multiuser mode
# 4 - unused
# 5 - X11
# 6 - reboot (Do NOT set initdefault to this)
```



# UNIX init

```
#
id:5:initdefault:

# System initialization.
si::sysinit:/etc/rc.d/rc.sysinit

10:0:wait:/etc/rc.d/rc 0
11:1:wait:/etc/rc.d/rc 1
12:2:wait:/etc/rc.d/rc 2
13:3:wait:/etc/rc.d/rc 3
14:4:wait:/etc/rc.d/rc 4
15:5:wait:/etc/rc.d/rc 5
16:6:wait:/etc/rc.d/rc 6
```



# UNIX/Linux init

```
# Trap CTRL-ALT-DELETE
ca::ctrlaltdel:/sbin/shutdown -t3 -r now

# When our UPS tells us power has failed, assume we have a few minutes
# of power left.  Schedule a shutdown for 2 minutes from now.
# This does, of course, assume you have powerd installed and your
# UPS connected and working correctly.
pf::powerfail:/sbin/shutdown -f -h +2 "Power Failure; System Shutting Down"
```





# UNIX/Linux init

```
# If power was restored before the shutdown kicked in, cancel it.  
pr:12345:powerokwait:/sbin/shutdown -c "Power Restored; Shutdown Cancelled"
```

```
# Run gettys in standard runlevels  
1:2345:respawn:/sbin/mingetty tty1  
2:2345:respawn:/sbin/mingetty tty2  
3:2345:respawn:/sbin/mingetty tty3  
4:2345:respawn:/sbin/mingetty tty4  
5:2345:respawn:/sbin/mingetty tty5  
6:2345:respawn:/sbin/mingetty tty6
```



# UNIX/Linux init

```
# Run xdm in runlevel 5  
x:5:respawn:/etc/X11/prefdm -nodaemon
```



# UNIX/Linux init

## ➤ Solaris 9 /etc/inittab

```
ap::sysinit:/sbin/autopush -f /etc/iu.ap
ap::sysinit:/sbin/soconfig -f /etc/sock2path
fs::sysinit:/sbin/rcS sysinit                >/dev/msglog 2<>/dev/msglog </dev/co
is:3:initdefault:
p3:s1234:powerfail:/usr/sbin/shutdown -y -i5 -g0 >/dev/msglog 2<>/dev/msglog
sS:s:wait:/sbin/rcS                          >/dev/msglog 2<>/dev/msglog </dev/co
s0:0:wait:/sbin/rc0                          >/dev/msglog 2<>/dev/msglog </dev/conso
```



# UNIX/Linux init

```
s1:1:respawn:/sbin/rc1          >/dev/msglog 2<>/dev/msglog </dev/conso
s2:23:wait:/sbin/rc2           >/dev/msglog 2<>/dev/msglog </dev/conso
s3:3:wait:/sbin/rc3            >/dev/msglog 2<>/dev/msglog </dev/conso
s5:5:wait:/sbin/rc5            >/dev/msglog 2<>/dev/msglog </dev/conso
s6:6:wait:/sbin/rc6            >/dev/msglog 2<>/dev/msglog </dev/conso
fw:0:wait:/sbin/uadmin 2 0      >/dev/msglog 2<>/dev/msglog </dev/conso
of:5:wait:/sbin/uadmin 2 6      >/dev/msglog 2<>/dev/msglog </dev/conso
rb:6:wait:/sbin/uadmin 2 1      >/dev/msglog 2<>/dev/msglog </dev/conso
co:234:respawn:/usr/lib/saf/ttymon -g -h -p "'uname -n' console login: " -T su
```



# UNIX bootup scripts

➤ Notice the location of the actual startup shell scripts can vary (`/etc`, `/etc/rc.d`, etc.), even between different versions of UNIX. For instance, Solaris: `/etc/rc?.d` (`/etc/rc2.d` is typical), CentOS Linux: `/etc/rc.d`



# UNIX/Linux Controlling init link sprawl

On some systems, you can use `chkconfig` to control the maze of links.

```
# chkconfig --list      # show what is on and what is off for runlevels
anacron                0:off          1:off          2:on           3:on           4:on
atd                    0:off          1:off          2:off          3:on           4:on
atievents             0:off          1:off          2:off          3:off          4:off
auditd                 0:off          1:off          2:on           3:on           4:on
autofs                 0:off          1:off          2:off          3:on           4:on

# chkconfig --level 2345 sendmail on      # have sendmail start 2345

# chkconfig --del sendmail                # remove sendmail from chkconfig management
# chkconfig --add sendmail                # add sendmail to chkconfig management
```



# Typical problems with UNIX booting

- ❏ Error in startup scripts (much more common problem on established servers than rather newly installed servers, and generally happens when a sysadmin has made some sort of change)
- ❏ The monitor/BIOS can't find bootblock or bootloader (much more likely during install)
- ❏ Kernel won't load or recognize your hardware (much



more likely during install)





# Typical problems with UNIX booting

- ☞ Can't find swap partition(s) (very much more likely during install)
- ☞ Damaged root (and/or /usr or maybe even /var) file system (not seen as often these days due to good journaling file systems such as ext3)



# Upshot

- ☞ You should know BASH syntax, so you can figure out what went wrong during boot.
- ☞ Occasionally you may wish to customize the shell scripts to either modify the “out of the box” behavior of the boot process or to add your own daemon startups.



# Upshot

- ☞ Traditionally, best practice has been widely considered to keep all of your local customizations in separate shell scripts (e.g., `/etc/rc.local` is a good place on Linux machines, `/etc/rc.d/S99local` Solaris). However, with systems governed by `chkconfig`; on those, it's best to use its scheme.



# Upshot

- ☞ Beware that many versions of UNIX/Linux choose to use symbolic links between a common script directory and the particular runlevel directory. Treat any modifications to the startup shell scripts as you would any other program – edit, test (reboot), document, debug until it works.



# Upshot

- 👉 In a major site, you may find that weekly reboots at an off time (such as early on Sunday morning) are automatically done to discover any bad interactions among boot modifications that might have been made recently.



# Linux booting

- ☞ Linux provides a flexible multi-operating system boot loader named grub (GRand Unified Bootloader) that can be used to boot different operating systems. Like Windows bootloaders, it can also sit on the Master Boot Record (MBR) of the boot device and transfer control to specific operating system kernels depending on either a user type-in or a default. Grub's behavior is controlled by `/boot/grub/grub.conf`.



☞ Previously, there was `lilo`, which is still seen occasionally, and even older, the `loadlin` program which used DOS to boot Linux.

☞ Here's a `grub` configuration file on a CentOS 3.6 machine with a choice of three different kernels:

```
# grub.conf generated by anaconda
#
# Note that you do not have to rerun grub after making c
# NOTICE:  You have a /boot partition.  This means that
#           all kernel and initrd paths are relative to /
#           root (hd0,0)
#           kernel /vmlinuz-version ro root=/dev/hda2
```



```
#          initrd /initrd-version.img
#boot=/dev/hda
default=0
timeout=10
splashimage=(hd0,0)/grub/splash.xpm.gz
title CentOS (2.4.21-37.EL)
    root (hd0,0)
    kernel /vmlinuz-2.4.21-37.EL ro root=LABEL=/
    initrd /initrd-2.4.21-37.EL.img
title CentOS (2.4.21-32.0.1.EL)
    root (hd0,0)
    kernel /vmlinuz-2.4.21-32.0.1.EL ro root=LABEL=/
    initrd /initrd-2.4.21-32.0.1.EL.img
```





```
title CentOS-3 (2.4.21-27.0.1.EL)
    root (hd0,0)
    kernel /vmlinuz-2.4.21-27.0.1.EL ro root=LABEL=/
    initrd /initrd-2.4.21-27.0.1.EL.img
```

- ☞ Since the boot loading process is reasonably common to various platforms, many boot loaders can handle different operating systems. You can, for instance, use grub to boot up Linux, Win2K, WinXP (and occasionally Vista, though by default Vista is not going to like a non-Windows MBR), and so forth; you can also use some third-party software, such as Neosmart



EasyBCD. (The last works particularly well with Vista and Linux dual-booting.)

- ❏ None of these are perfect. The ancient Partition Magic usually works for XP/2003, but there are many reports of issues with Vista; Neosmart is a very good choice for Vista/Linux dualboot.

