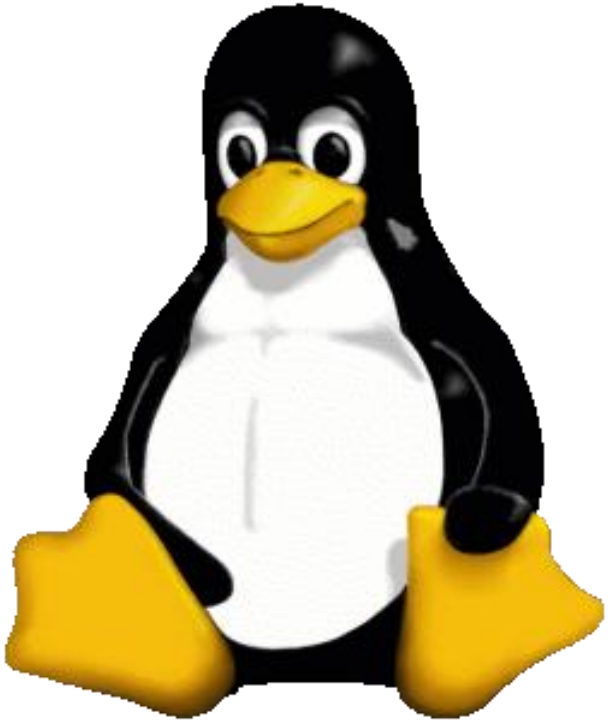
Chapter 4: Booting and Kernel Initialization

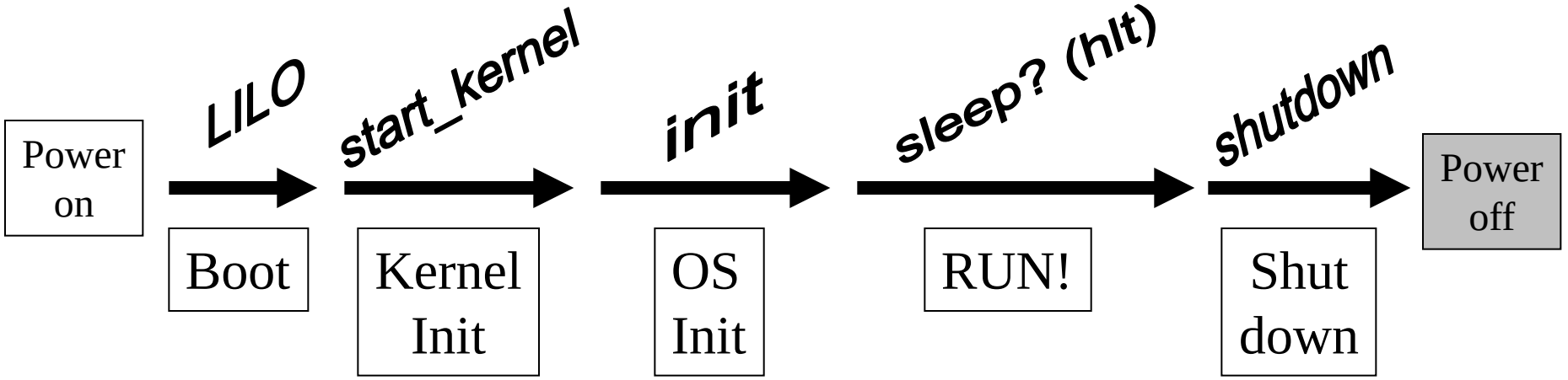


Objectives

- Explain **system lifecycle** from power-on to power-off.
- Describe general principles involved in **booting** a system and specific details of a standard LILO disk-based boot on the Intel architecture.
- Motivate and clarify the **transfer of control** from hardware, to firmware, to software during system boot.
- Trace significant events in **kernel initialization**.
- Demonstrate role and importance of the **init** process.
- Review **shutdown** procedures.
- Briefly survey a variety of **advanced boot concepts**.
- Briefly consider **power management** issues.

System Lifecycle: Ups & Downs

- Booting
- Kernel Initialization
- init: Process Number One
- Shutdown
- Advanced Boot Concepts
- Power Management

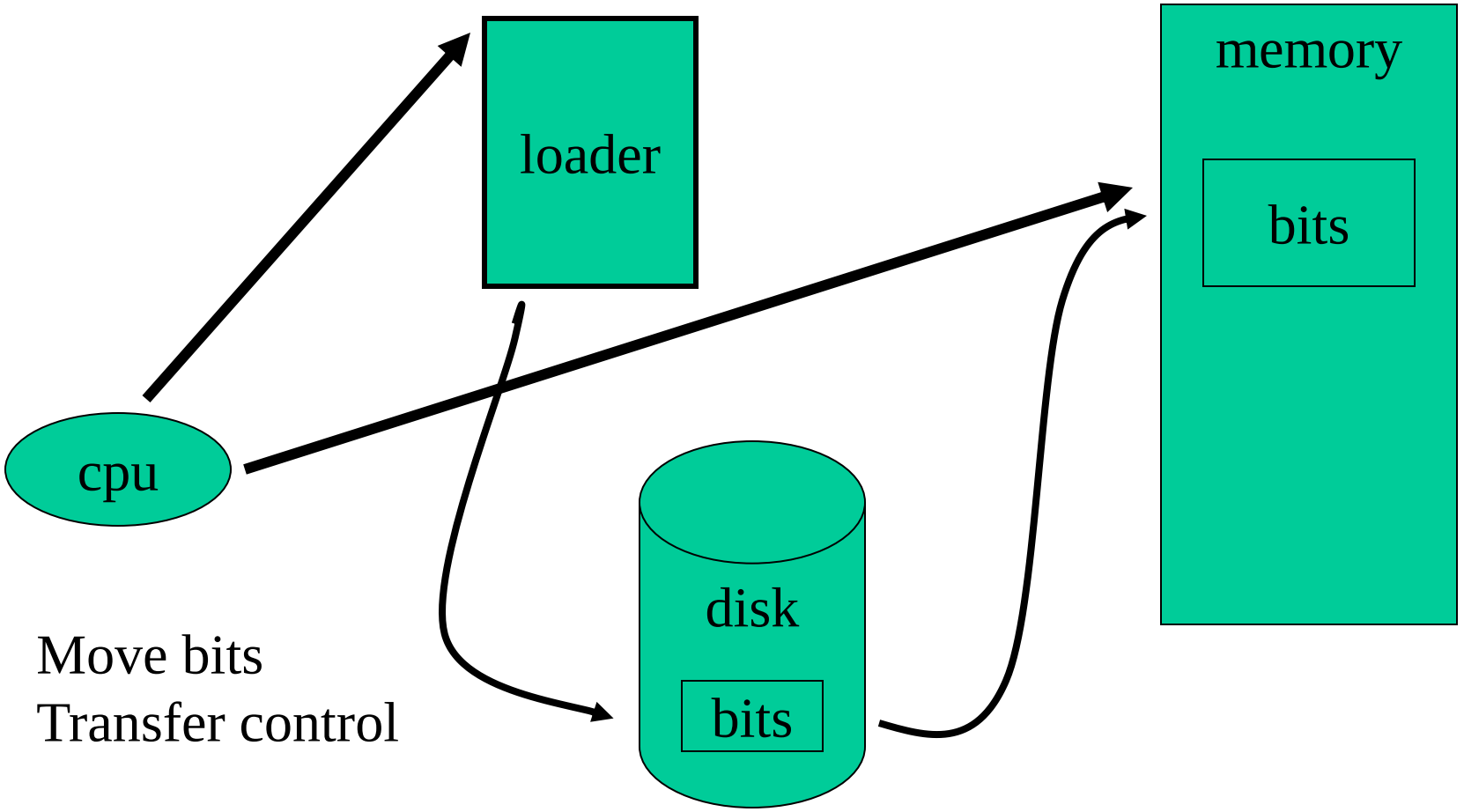


Boot Terminology

- Loader
  - Program that moves bits from disk (usually) to memory and then transfers CPU control to the newly "loaded" bits (executable)
- Bootloader / Bootstrap
  - Program that loads the "first program" (the kernel)
- Boot PROM / PROM Monitor / BIOS
  - Persistent code that is "already loaded" on power-up
- Boot Manager
  - Program that lets you choose the "first program" to load

What's a Loader?

- A program that moves bits (usually) from disk to memory and then transfers control to the newly loaded bits (executable).

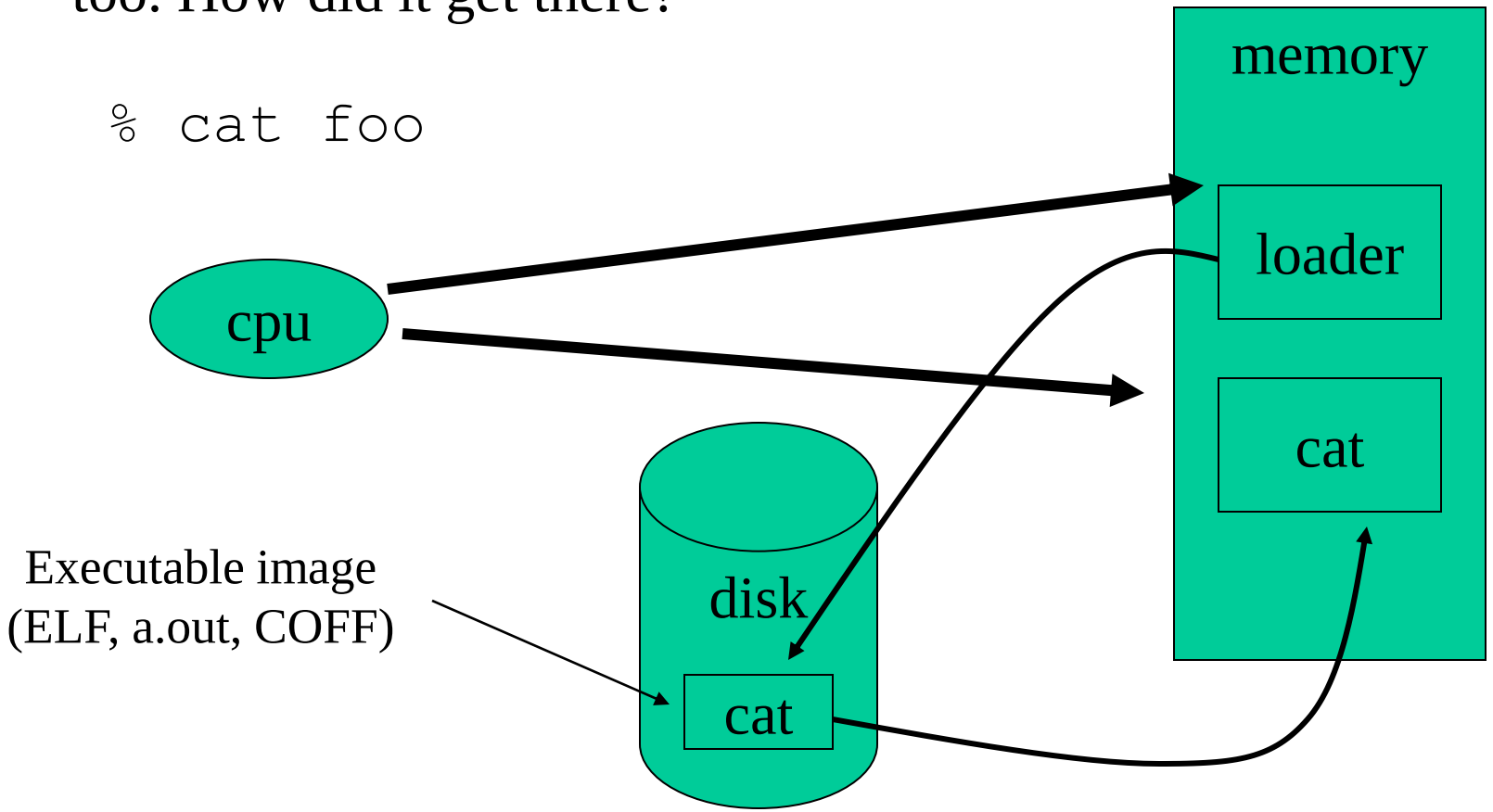


1) Move bits
2) Transfer control

Who Loads the Loader?

- Of course, the loader is just a program and it resides in memory too. How did it get there?

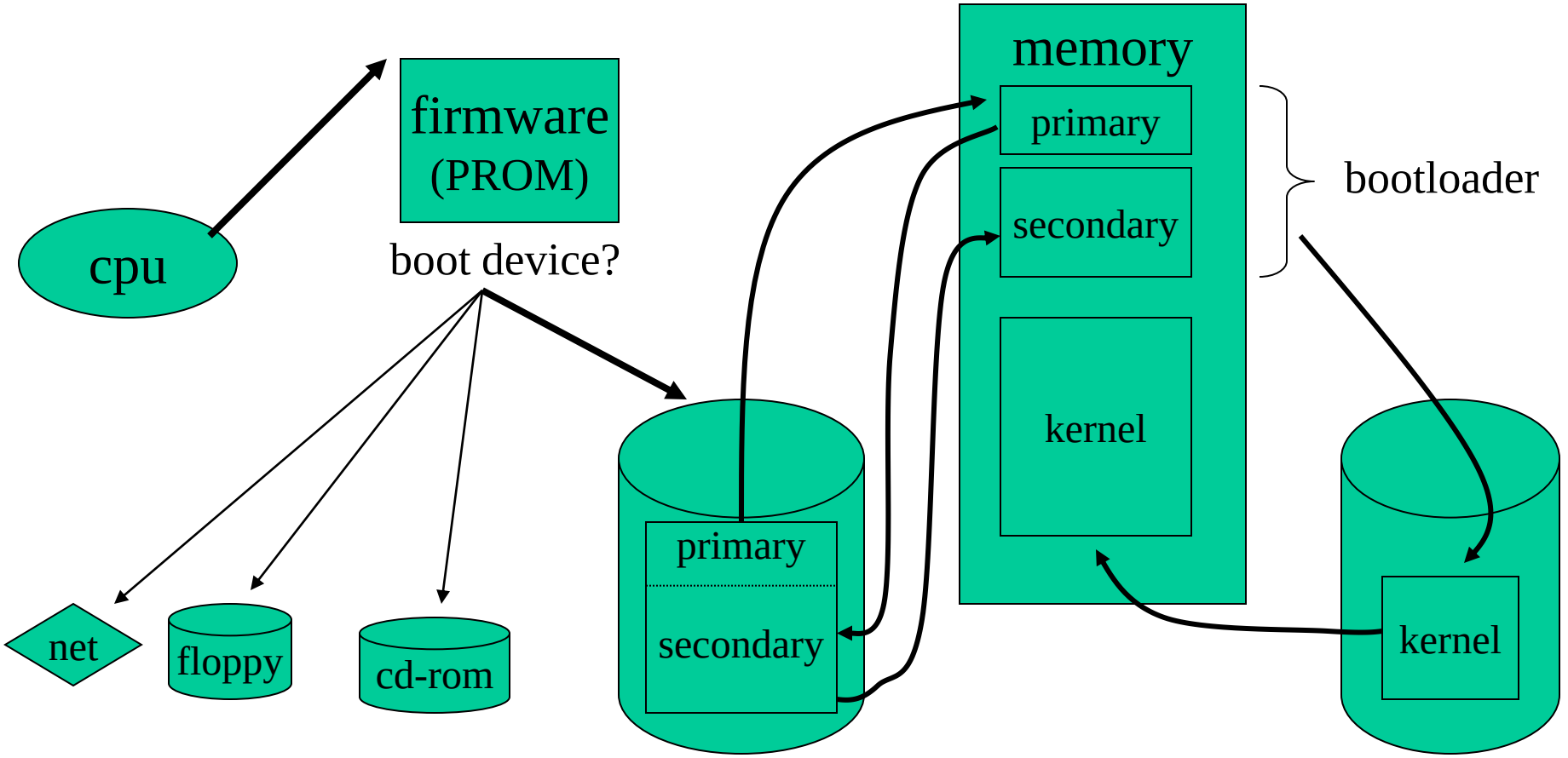
```
% cat foo
```



- We need a “loader loader” …

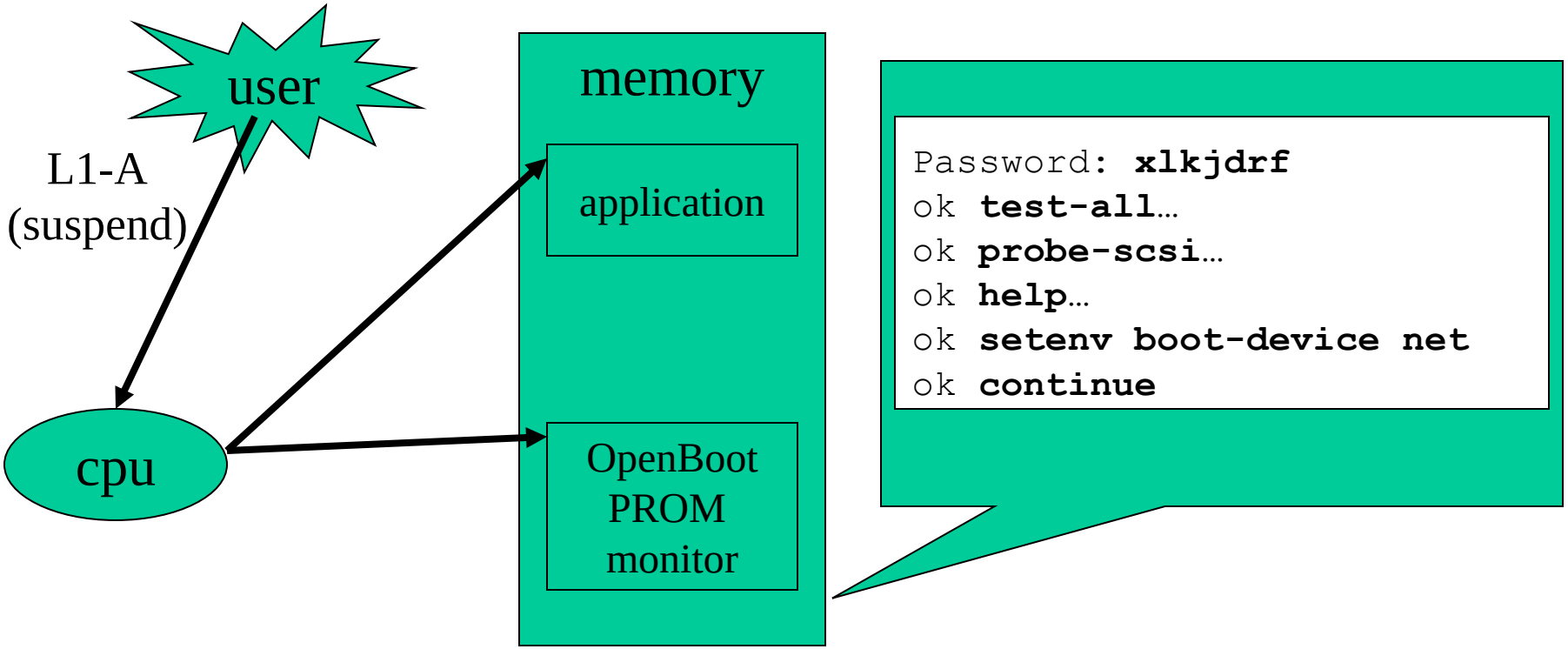
Bootstrap Loader (Bootloader)

- The program that loads the “first program”
- Usually “staged”: primary, secondary
- Requires firmware support (“hardware bootstrap”)



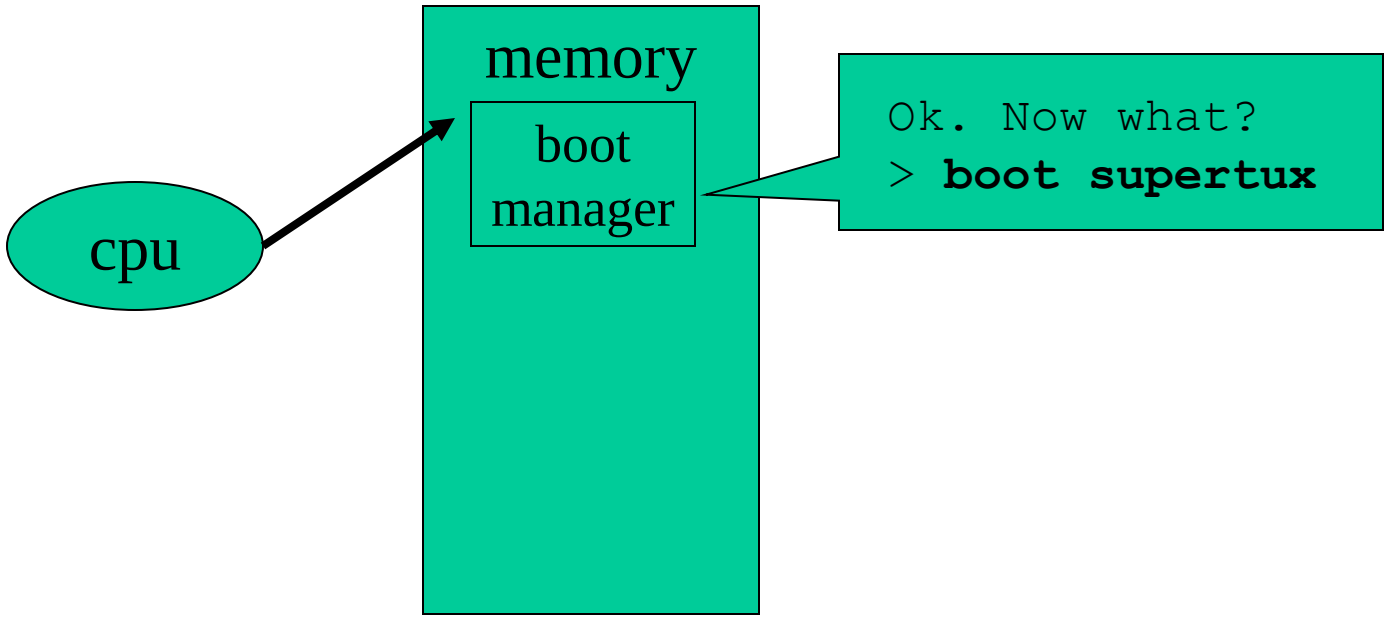
PROM Monitors vs. BIOS

- BIOS: limited setup via DEL or F1 at boot
- Monitor: continuously accessible command interpreter
- Examples: Sparc OpenBoot, Alpha SRM



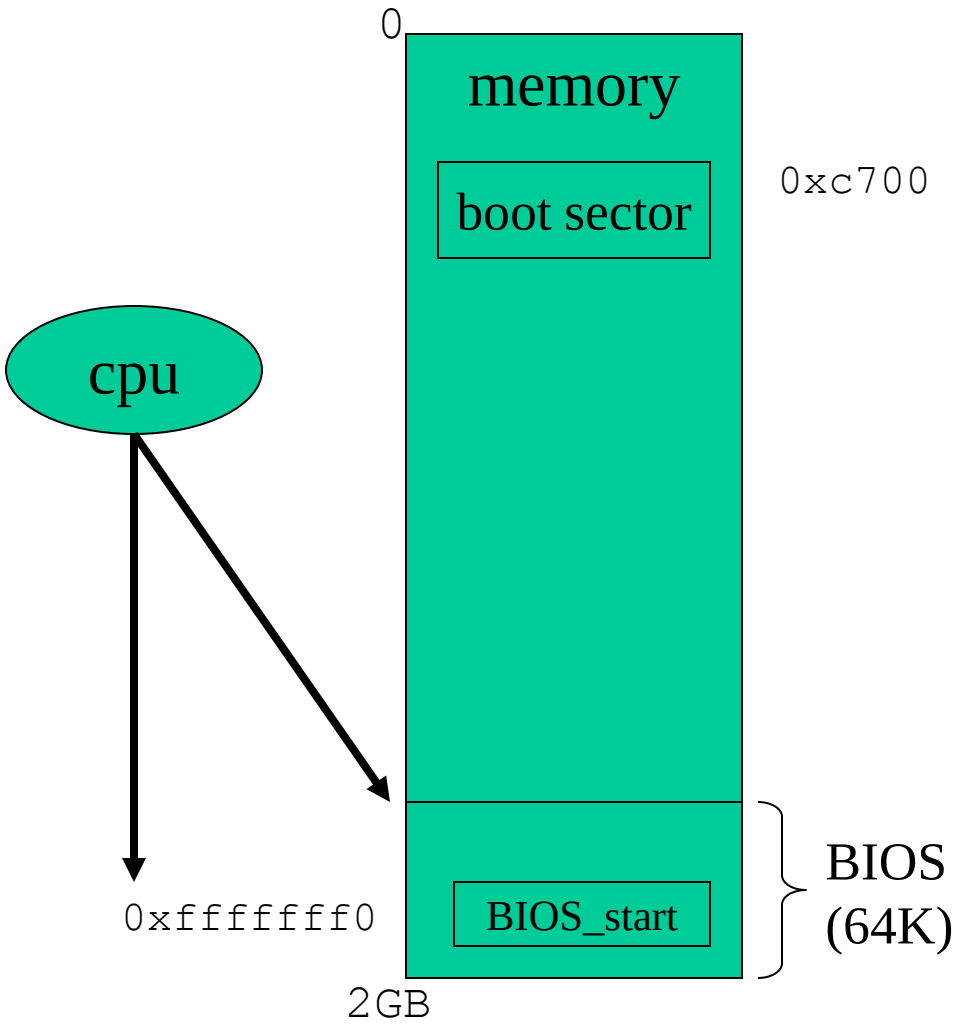
Boot Managers

- Code loaded by firmware bootstrap that allows choice of boot image, specification of boot parameters, etc.
- Adds another “layer” to boot process but increases flexibility, supports “multiboot” configurations
- Examples: LILO, System Commander



Booting the PC

- Intel X86 firmware loads a 512 byte “boot sector” at 0x7C00 and transfers control in real-mode (640K limit)



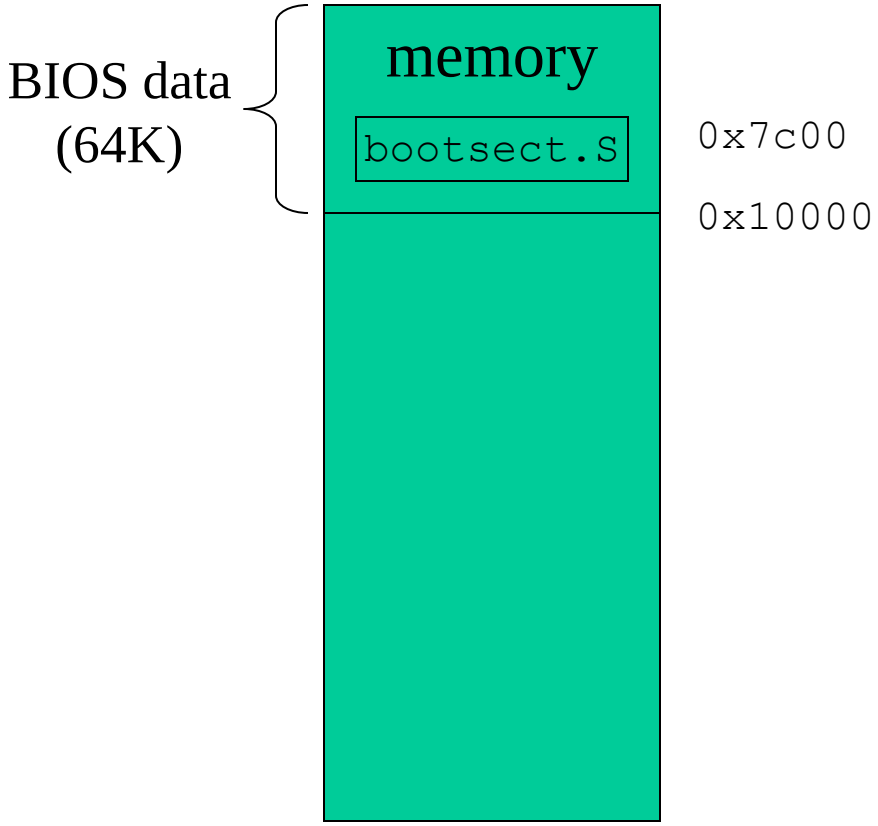
1. Power On Self Test (POST)
2. Generate INT 19h (bootstrap)
3. Select boot device
4. Load boot sector
    1. floppy: first sector
    2. hard disk: MBR (mboot) or partition boot block (pboot)
5. Verify “magic number”
6. Execute boot sector (primary bootloader)

Booting from a Floppy 1

- zImage: compressed kernel dumped directly to floppy
- Boot complicated by real-mode, BIOS, compression

1

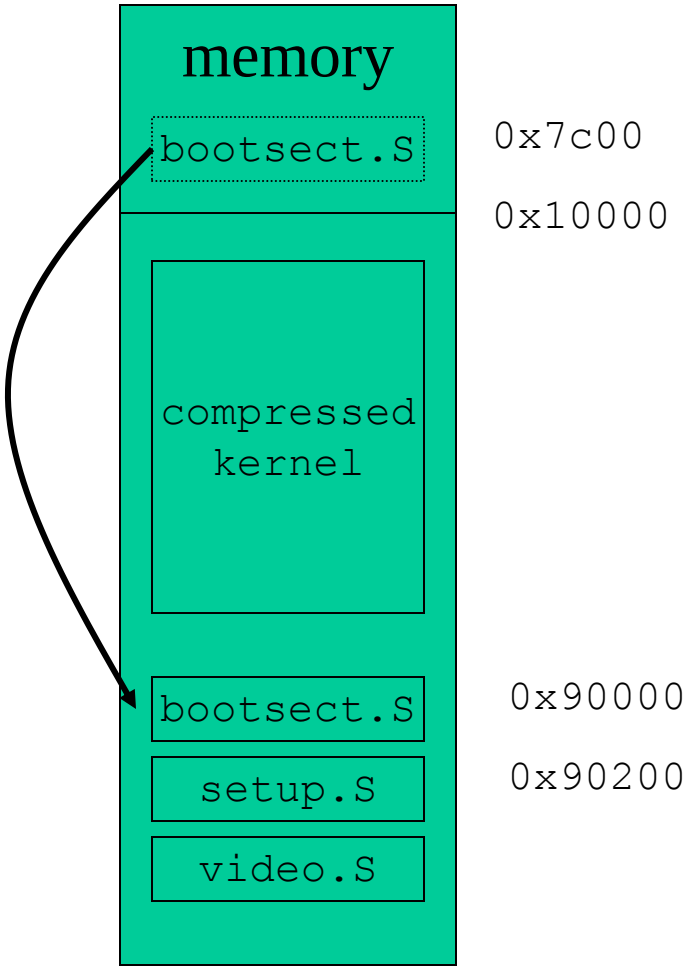
- BIOS loads boot sector
- Transfers control



Booting from a Floppy 2

2

- Boot sector moves itself (!) to 0x90000
- Loads two more sectors at 0x90200
  - arch/i386/boot/setup.S
  - arch/i386/boot/video.S
- Loads compressed kernel after BIOS data
- Transfers control to setup.S
- Performs real-mode hardware init



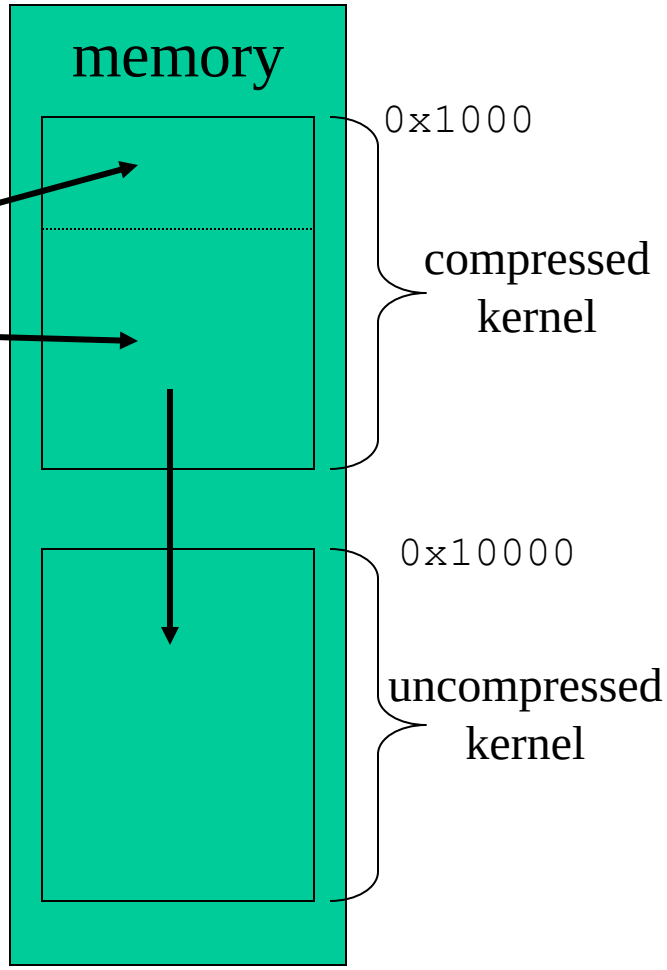
Booting from a Floppy 3

3

- setup.S copies kernel to 0x1000 (4K)
- Avoids wasted space but trashes BIOS data
- Kernel consists of two pieces:
  - arch/i386/boot/compressed/head.S
  - arch/i386/kernel/head.S

- Enters **protected mode**

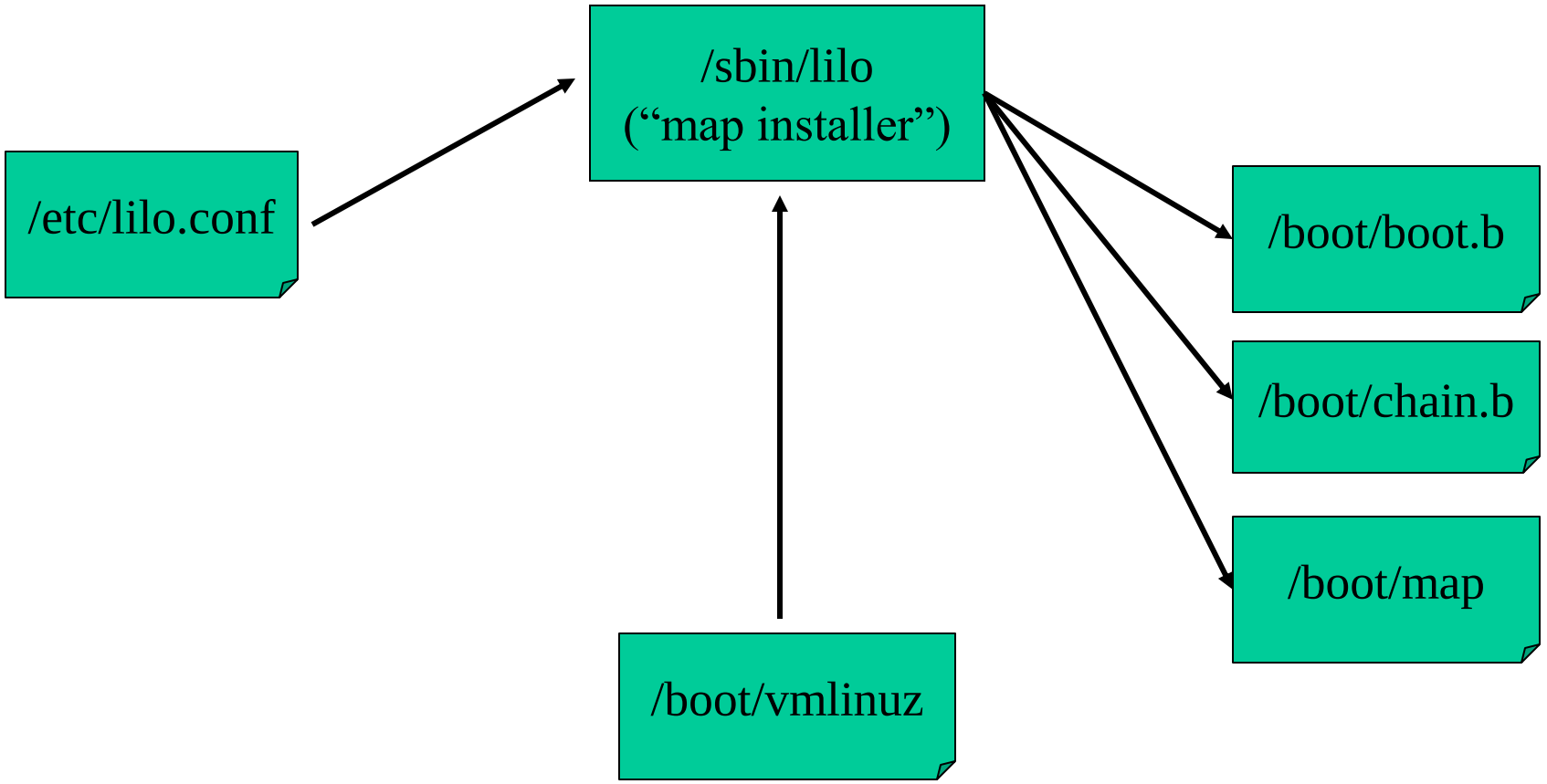
- Jumps to 0x1000 (compressed/head.S)
- compressed/head.S
  - Sets up stack
  - “Self-extracts” kernel
  - Jumps to 0x10000 (kernel/head.S)
- kernel/head.S sets up paging
- Jumps to init/main.c:start_kernel (!)



LILO: LInux LOader

- A versatile boot manager that supports:
  - Choice of Linux kernels
  - Boot time kernel parameters
  - Booting non-Linux kernels
  - A bewildering variety of configurations
- Characteristics:
  - Lives in MBR or partition boot sector
  - Has no knowledge of filesystem structure so…
  - Builds a sector “map file” (block map) to find kernel
- /sbin/lilo – “map installer”
  - Builds map file, boot sector
  - Run after change to kernel or /etc/lilo.conf

LILO Components



Example lilo.conf File

```
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
prompt
timeout=50
default=linux

image=/boot/vmlinuz-2.2.12-20
        label=linux
        initrd=/boot/initrd-2.2.12-20.img
        read-only
        root=/dev/hda1
```

Booting from Disk with LILO

- LILO prints a progress string "LILO boot:"

1. BIOS loads boot sector at 0x7c00
   Moves itself to 0x9a00 L
2. Sets up stack
   Loads secondary bootloader at 0x9b00 I
3. Transfers control to secondary L
4. Loads "block map" at 0x9d200
   Loads default command line at 0x9d600 ..... O
5. Waits for user input or timeout ..... boot:

Skiff Bootloader Startup

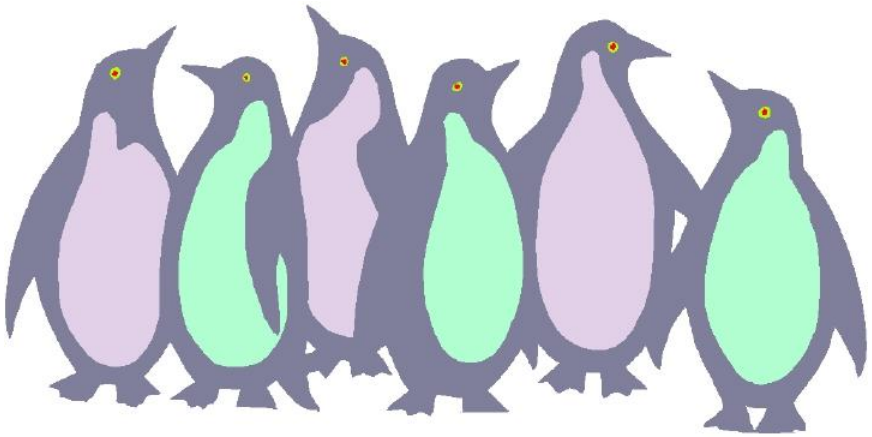

```
10000000
20000000
40000000
80000000
ENDM
[[ text removed ]]
>> Compaq Personal Server BootLoader, Rev 2.00(a.out)
>> (Edwin Foo,Jamey Hicks,Dave Panariti,Mike Schexnaydre,Chris Joerg), 00-01-07_15:36
>>  SA110 Rev=4401A103
>> (c) 1999 Compaq Cambridge Research Laboratory
Press Return to start the OS now, any other key for monitor menu
eval param blk
+ set dram_size 02000000
  setting param=dram_size to value=02000000
+ set hostname skiff137
  setting param=hostname to value=skiff137
+ set ipaddr 192.168.0.3
  setting param=ipaddr to value=192.168.0.3
+ set gateway 192.168.0.1
  setting param=gateway to value=192.168.0.1
+ set netmask 255.255.255.0
  setting param=netmask to value=255.255.255.0
+ set nfs_server_address 192.168.0.2
  setting param=nfs_server_address to value=192.168.0.2
```

Skiff Bootloader Commands

Skiff Bootloader has commands for:

- managing non-volatile parameter storage,
- selecting a boot target,
- loading flash or ram via XMODEM over the serial connection,
- and examining system configurations and memory.

```
boot> help
Available Commands:
? | help
help <command> | <command> help
boot [flash|ram|net|nonet|alt]
load [[flash|ram <dstaddr>] | kernel | bootldr | params | usercode]
peek ram|flash|int|short|byte <addr>
poke ram|flash|int|short|byte <addr>
qflash [cfi|autoselect] <waddr>
eflash <sectoraddr>|chip
physaddr <vaddr> -- returns <paddr>
set <param> <value>
show [<param>]
evalparams
params [eval|show|save|reset]
boot>
```



Skiff Bootloader Parameters

```
boot> params show
  os=autoselect
  boot_type=flash
  force_unzip=0x00000000
  entry=0x10000000
  ramdisk=0x00000000
  dram_size=0x00080000
  dcache_enabled=0x00000001
  icache_enabled=0x00000001
  memc_ctrl_reg=0x0000110C
  maclsbyte=0x000000FF
  serial_number=0x000000FF
  system_rev=0x00000002

  linuxargs= root=/dev/ram initrd ramdisk_size=8192

  hostname=skiff137
  ipaddr=192.168.0.3
  gateway=192.168.0.1
  netmask=255.255.255.0
  nfs_server_address=192.168.0.2
  nfsroot=

  noerase=0x00000000
  xmodem=0x00000001
```

Kernel boot params

Network params

start_kernel

- **`init/main.c:start_kernel`**

identify bootstrap processor (BSP)
setup_arch()
init crucial subsystems
parse_options()
setup kernel profiling
enable interrupts (sti())
calibrate_delay() -- BogoMIPS
init subsystems needing delay
check_bugs()
smp_init()
spawn init as a “kernel thread”

become idle process!

BogoMIPS

- BogoMIPS is roughly the number of times per second the CPU can execute a short delay loop
- Used by device init code for short waits
- Widely misused to measure performance

- **`init/main.c:`**`calibrate_delay`

wait for next clock tick (jiffie)

make initial estimate

verify estimate, adjusting as necessary

print BogoMIPS (without using floating-point!)

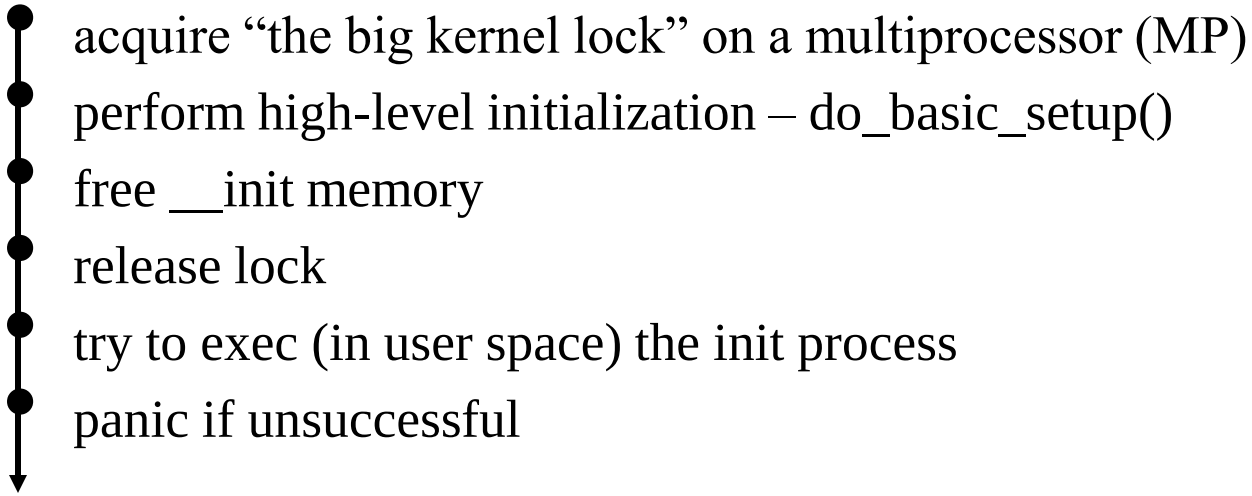
Kernel Options

- Linux accepts a large number of command line options (see BootPrompt-HOWTO)
- parse_options() parses and acts on some; the rest are passed to the init process as arguments
- Examples:
  - `debug ro rw initrd= noinitrd ramdisk= profile= reboot=`
  - `init=/some/other/program swap= mem= nfsroot=`
  - lots of driver-specific options

init()

- init() begins life as a “kernel thread” and ends by starting the user-level init process (/sbin/init)

- **`init/main.c:init`**



do_basic_setup()

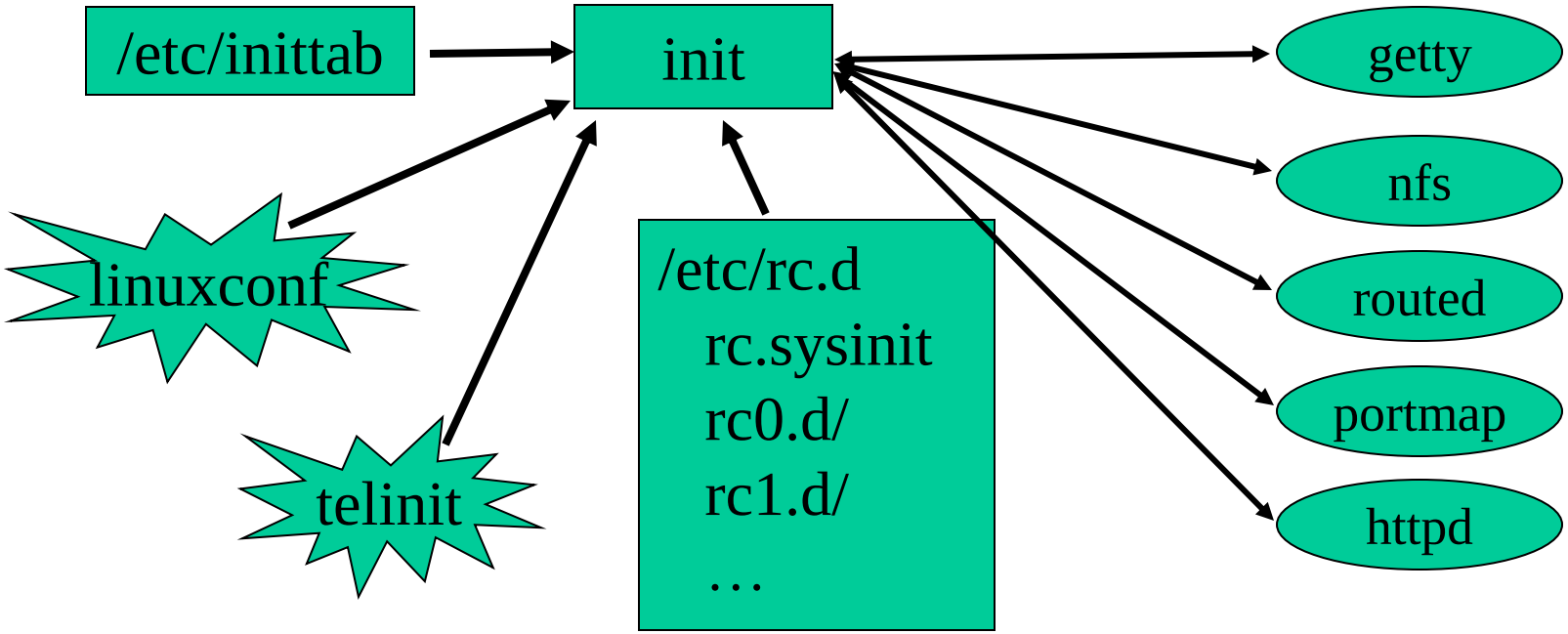
- Perform “high-level” initialization requiring memory and process management to be setup

- **`init/main.c:`**`do_basic_setup`

- do conditional bus init (pci, sbus, mca, etc.)
- sock_init()
- spawn update (bdflush), paging (kpiod) and swapping (kswapd) threads
- device_setup()
- filesystem_setup()
- binfmt_setup()
- mount_root()
- conditionally execute /linuxrc from “initial ramdisk” (initrd)

/sbin/init

- Ancestor of all processes (but idle); “reaps” children
- Controls transitions between “runlevels”
  - 0: shutdown 1: single-user 2: multi-user (no NFS)
  - 3: full multi-user 5: X11 6: reboot
- Executes startup/shutdown scripts for each runlevel

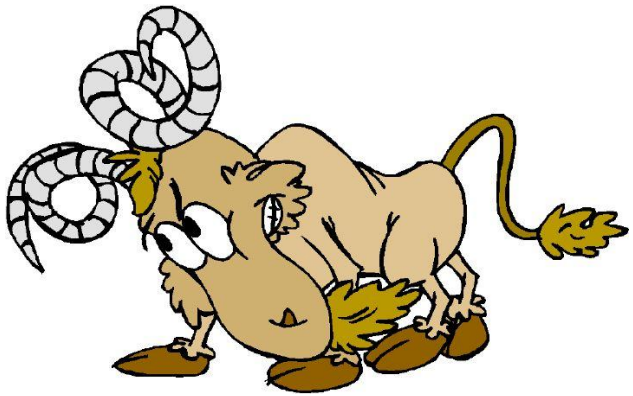


Shutdown

- Linux buffers writes; use /bin/shutdown to avoid data loss and filesystem corruption
- shutdown inhibits login, asks init to send SIGTERM to all processes, then SIGKILL
- Low-level commands: halt, reboot, poweroff
  - use -h, -r or -p options to shutdown instead
- Ctrl-Alt-Delete “Vulcan neck pinch”
  - defined by a line in /etc/inittab
  - ca::ctrlaltdel:/sbin/shutdown -t3 -r now

Advanced Boot Concepts

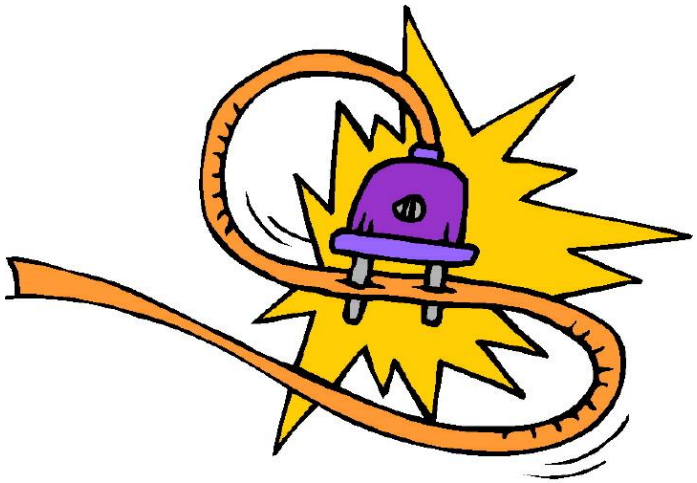
- Initial ramdisk (initrd) – two-stage boot for flexibility
  - first mount “initial” ramdisk as root
  - execute /linuxrc to perform additional setup, configuration
  - then mount “real” root and continue
  - see Documentation/initrd.txt for details
  - also see “man initrd”



- Net booting:
  - remote root (Diskless-root-HOWTO)
  - diskless boot (Diskless-HOWTO)

- RAID root – tricks for high-performance root
- OpenPROM – open-source BIOS; burn your own!

Power Management



- Halting in the idle process
  - idle process executes hlt on Intel
  - low-power consumption mode
- Suspending the system
  - patches for suspending to disk
- APM: Advance Power Management
  - laptop standard power management
- ACPI: Advanced Configuration and Power Interface
  - new comprehensive standard from Intel-Microsoft
- Power-management is essential for mobile systems

Summary

- Bootstrapping a system is a complex, device-dependent process that involves transition from hardware, to firmware, to software.
- Booting within the constraints of the Intel architecture is especially complex and usually involves firmware support (BIOS) and a boot manager (LILO).
- /sbin/lilo is a “map installer” that reads configuration information and writes a boot sector and block map files used during boot.
- The Skiff bootloader manages non-volatile memory, loads flash memory and ram, and allows selection of boot targets.
- start_kernel is Linux “main” and sets up process context before spawning process 0 (idle) and process 1 (init).
- The init() function performs high-level initialization before exec’ing the user-level init process.

Booting and Kernel Initialization Lab