LINUX KERNEL

PRABHAT RANJAN

(email : prabhat_ranjan@daiict.ac.in)

OUTLINE

- Different states of kernel
- Directory structure of kernel source
- Description of various directory
- “proc” file system
- Kernel Compilation
- Resources

What is the kernel?

The kernel is the "core" of any computer system: it is the "software", which allows users to share computer resources.

The kernel can be thought of as the main software of the OS (Operating System), which may also include graphics management.

For example, under Linux (like other Unix-like OSs), the XWindow environment doesn't belong to the Linux Kernel, because it manages only graphical operations (it uses user mode I/O to access video card devices). By contrast, Windows environments (Win9x and so on) are a mix between a graphical environment and kernel.

Difference between User Mode and Kernel Mode?

Many years ago, when computers were as big as a room, users ran their applications with much difficulty and, sometimes, their applications crashed the computer.

To avoid having applications that constantly crashed, newer OSs were designed with 2 different operative modes:

Kernel Mode: the machine operates with critical data structure, direct hardware (IN/OUT or memory mapped), direct memory, IRQ, DMA, and so on.

User Mode: users can run applications.

Kernel Mode "prevents" User Mode applications from damaging the system or its features.

Modern microprocessors implement in hardware at least 2 different states. For example under Intel, 4 states determine the PL (Privilege Level). It is possible to use 0,1,2,3 states, with 0 used in Kernel Mode.

Unix OS requires only 2 privilege levels.

Switching from User Mode to Kernel Mode

Once we understand that there are 2 different modes, we have to know when we switch from one to the other.

Typically, there are 2 points of switching:

1. When calling a System Call: after calling a System Call, the task voluntarily calls pieces of code living in Kernel Mode

2. When an IRQ (or exception) comes: after the IRQ, an IRQ handler (or exception handler) is called, then control returns back to the task that was interrupted like nothing had happened.

System Calls

System calls are like special functions that manage OS routines which live in Kernel Mode.

A system call can be called when we:

- access an I/O device or a file (like read or write)
- need to access privileged information (like pid, changing scheduling policy or other information)
- need to change execution context (like forking or executing some other application)
- need to execute a particular command (like "chdir", "kill" or "signal")

System calls are almost the only interface used by User Mode to talk with low level resources (hardware).

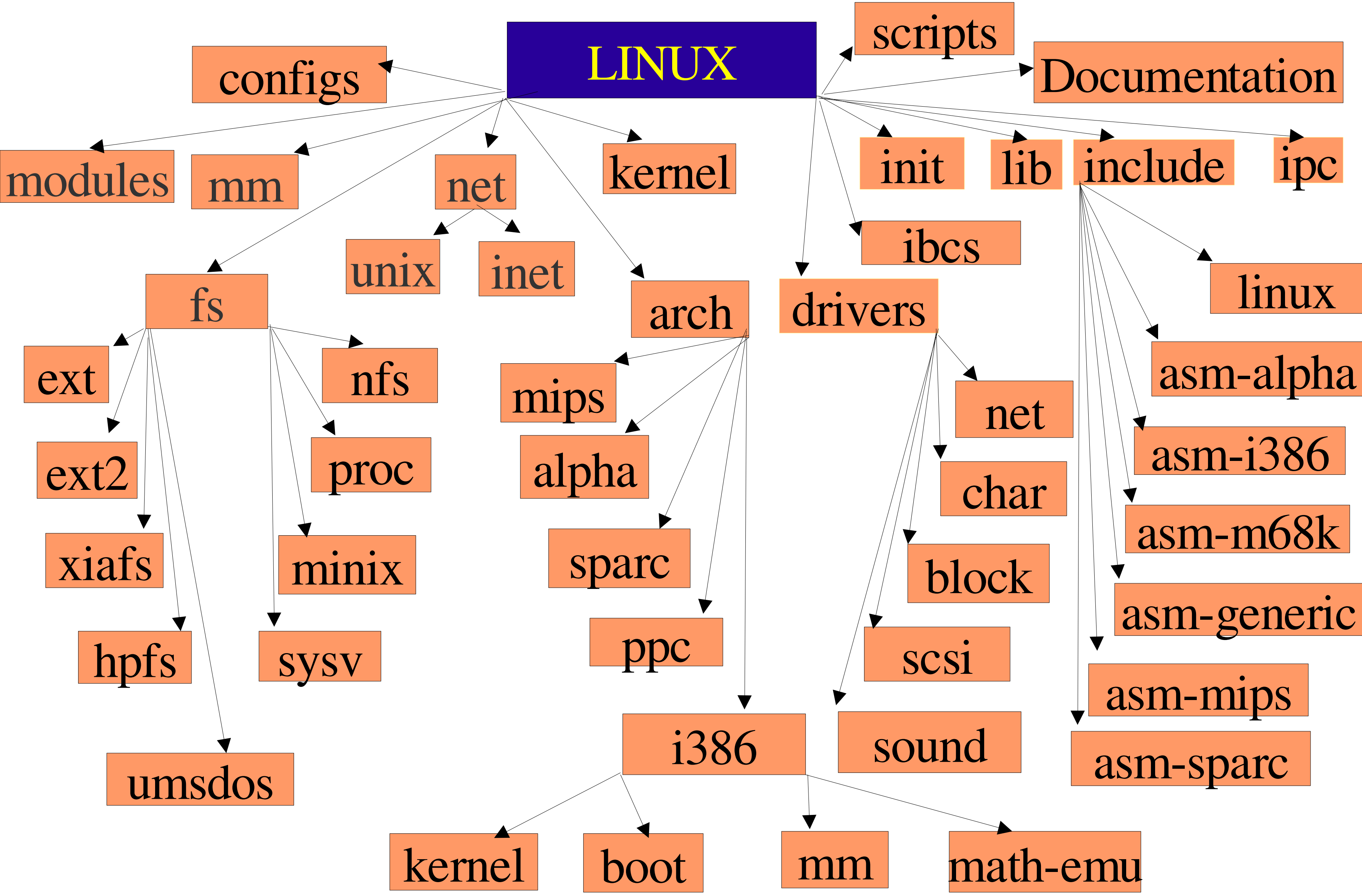
The only exception to this statement is when a process uses "ioperm" system call.

In this case a device can be accessed directly by User Mode process (IRQs cannot be used).

NOTE: Not every "C" function is a system call, only some of them.

arch/i386/kernel/entry.S

sys_call_table



The directory structure of the LINUX sources

arch/<-------------> init/

arch/ - Processor architecture dependent files

arch/i386/boot - assembler sources to take care of initial booting

=> load kernel

=> installs ISRs

=> ...

init/ - functions needed to start the kernel

=> function start_kernel( ), which initializes the kernel correctly taking account of boot parameters passed to it

=> the first process `init' is created

kernel/<-----------> arch/i386/kernel

kernel/

arch/i386/kernel - central sections of kernel

=> main system calls fork, exit

=> time management (system time, timers..),

=> the scheduler,

=> the DMA, IRQ management,

=> signal handling

mm -> memory management

mm/

arch/i386/mm - memory management sources

=> requesting and releasing system kernel memory

=> saving unused pages of memory to hard disk (paging)

=> inserting file and memory areas at specified addresses (mmap)

=> virtual memory interface

=> signal handling

fs-> file systems

fs/ - various file systems

fs/ext2 - the now standard linux file system

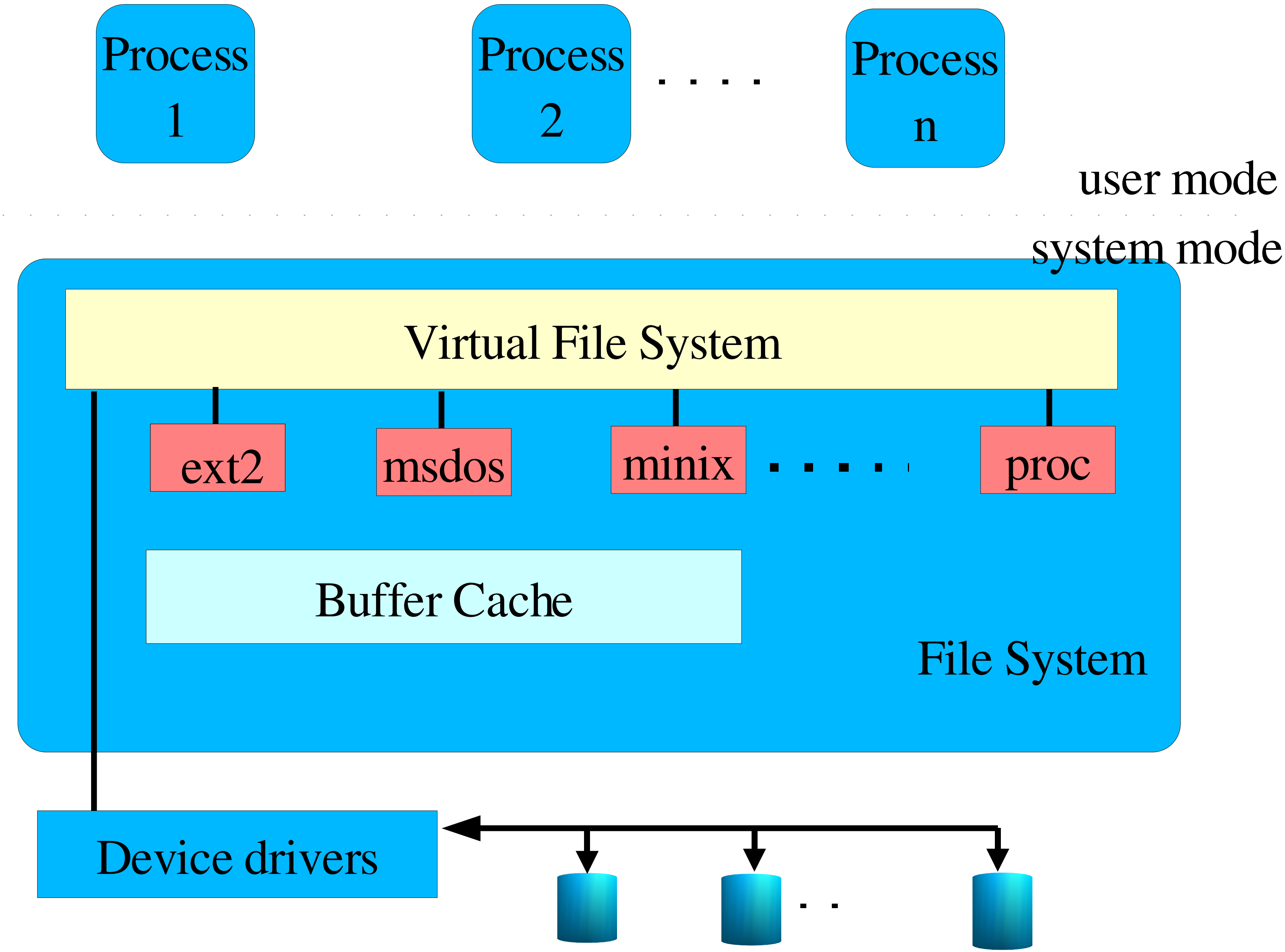
fs/proc - current status of linux kernel and running processes

Virtual File System

supplies the applications with the system calls for file management

maintains internal structure and passes task to the appropriate actual file system

VFS performs standard jobs, such as lseek( ), which is not provided by actual file system



Process 1
Process 2
. . . .
Process n
user mode
system mode
Virtual File System
ext2
msdos
minix
proc
Buffer Cache
File System
Device drivers

device drivers

drivers/block/ - the device drivers for block-oriented hardware,

drivers/cdrom/ - the device drivers for proprietary CD -ROM drives.

drivers/char/ - the driver for character-oriented devices,

drivers/isdn/ - the ISDN drivers,

drivers/net/ - the drivers for various network cards,

drivers/pci/ - PCI bus access and control,

drivers/sbus/ - access and control of Sparc machines' S buses,

drivers/scsi/ - the SCSI interface, and

drivers/sound/ - the sound card drivers.

Other Directories

ipc/ - classical interprocess communication as (IPC) as per system V. These include semaphores, shared memory and message queues

net/ - the implementation of various network protocols (TCP/IP, ARP etc.) and the code for sockets to the UNIX and internet domains

lib/ - some standard C library functions to provide for programming of kernel as per C conventions

module/ - modules generated when the kernel is compiled, no source code

include/ - kernel-specific header files, very important for kernel programming

/proc file System

The proc file system acts as an interface to internal data structures in the kernel.

It can be used to obtain information about the system and to change certain kernel parameters at runtime (sysctl).

First, we'll take a look at the read-only parts of /proc. Later, we see how we can use /proc/sys to change kernel parameter settings.

The directory /proc contains (among other things) one subdirectory for each process running on the system, which is named after the process ID (PID).

The link self points to the process reading the file system.

Each process subdirectory has the entries listed in Table 1-1.

Table : Process specific entries in /proc

File	Content	
cmdline	Command line arguments	
cpu	Current and last cpu in wich it was executed	(2.4)(smp)
cwd	Link to the current working directory	
environ	Values of environment variables	
exe	Link to the executable of this process	
fd	Directory, which contains all file descriptors	
maps	Memory maps to executables and library files	(2.4)
mem	Memory held by this process	
root	Link to the root directory of this process	
stat	Process status	
statm	Process memory status information	
status	Process status in human readable form	

Similar to the process entries, the kernel data files give information about the running kernel.

The files used to obtain this information are contained in /proc and are listed in Table.

Not all of these will be present in your system. It depends on the kernel configuration and the loaded modules, which files are there, and which are missing.

Table : Kernel info in /proc

File	Content	
apm	Advanced power management info	
bus	Directory containing bus specific information	
cmdline	Kernel command line	
cpuinfo	Info about the CPU	
devices	Available devices (block and character)	
dma	Used DMS channels	
filesystems	Supported filesystems	
driver	Various drivers grouped here, currently rtc	(2.4)
execdomains	Execdomains, related to security	(2.4)
fb	Frame Buffer devices	(2.4)
fs	File system parameters, currently nfs/exports	(2.4)
ide	Directory containing info about the IDE subsystem	

Table 1-3: Kernel info in /proc

interrupts	Interrupt usage	
iomem	Memory map	(2.4)
ioports	I/O port usage	
irq	Masks for irq to cpu affinity	(2.4)(smp?)
isapnp	ISA PnP (Plug&Play) Info	(2.4)
kcore	Kernel core image (can be ELF or A.OUT(deprecated in 2.4))	
kmsg	Kernel messages	
ksyms	Kernel symbol table	
loadavg	Load average of last 1, 5 & 15 minutes	
locks	Kernel locks	
meminfo	Memory info	
misc	Miscellaneous	
modules	List of loaded modules	

Table 1-3: Kernel info in /proc

mounts	Mounted filesystems
net	Networking info (see text)
partitions	Table of partitions known to the system
pci	Depreciated info of PCI bus (new way -> /proc/bus/pci/, decoupled by lspci (2.4)
rtc	Real time clock
scsi	SCSI info (see text)
stat	Overall statistics
swaps	Swap space utilization
sys	See chapter 2
sysvipc	Info of SysVIPC Resources(msg, sem, shm)(2.4)
tty	Info of tty drivers
uptime	System uptime
version	Kernel version

* Modifying kernel parameters by writing into files found in /proc/sys

* Exploring the files which modify certain parameters

* Review of the /proc/sys file tree

A very interesting part of /proc is the directory /proc/sys.

This is not only a source of information, it also allows you to change parameters within the kernel.

Be very careful when attempting this. You can optimize your system, but you can also cause it to crash.

Never alter kernel parameters on a production system.

Set up a development machine and test to make sure that everything works the way you want it to.

You may have no alternative but to reboot the machine once an error has been made.

/proc/sys/fs - File system data

This subdirectory contains specific file system, file handle, inode, dentry and quota information.

dquot-nr and dquot-max

The file dquot-max shows the maximum number of cached disk quota entries.

The file dquot-nr shows the number of allocated disk quota entries and the number of free disk quota entries.

If the number of available cached disk quotas is very low and we have a large number of simultaneous system users, we might want to raise the limit

file-nr and file-max

The kernel allocates file handles dynamically, but doesn't free them again at this time.

The value in file-max denotes the maximum number of file handles that the Linux kernel will allocate. When you get a lot of error messages about running out of file handles, you might want to raise this limit.

```
# cat /proc/sys/fs/file-max
4096
# echo 8192 > /proc/sys/fs/file-max
# cat /proc/sys/fs/file-max
8192
```

```
# cat file-max
9824
# cat file-nr
1288    191    9824
```

The three values in file-nr denote the number of allocated file handles, the number of remaining file handles, and the maximum number of file handles.

When the allocated file handles come close to the maximum, but the number of actually used ones is far behind, you've encountered a peak in your usage of file handles and you don't need to increase the maximum.

inode-state, inode-nr and inode-max

As with file handles, the kernel allocates the inode structures dynamically, but can't free them yet.

The value in inode-max denotes the maximum number of inode handlers. This value should be 3 to 4 times larger than the value in file-max, since stdin, stdout, and network sockets also need an inode struct to handle them. If you regularly run out of inodes, you should increase this value.

SUMMARY

Certain aspects of kernel behavior can be modified at runtime, without the need to recompile the kernel, or even to reboot the system.

The files in the /proc/sys tree can not only be read, but also modified.

One can use the echo command to write value into these files, thereby changing the default settings of the kernel.

/usr/src/linux/Documentation/filesystems/proc.txt

sysctl => /etc/sysctl.conf

KERNEL COMPILATION

make config - makes step-by-step interrogation of various option - **tedious**
make menuconfig - ncurses based menu driven
make xconfig - tcl/tk based menu driven
make oldconfig

make depend
make clean
make bzImage
make modules
make modules_install

Assign Unique Name

- Inside the Linux source directory is the default Makefile. This file is used by the make utility to compile the Linux sources.
  - VERSION = 2
  - PATCHLEVEL = 4
  - SUBLEVEL = 22
  - EXTRAVERSION = -1

Assign Unique Name

- an additional EXTRAVERSION field.
- To prevent overwriting any existing kernel modules on the system change this EXTRAVERSION to something unique
- When the final installation steps are run, kernel module files will then get written to /lib/modules/$VERSION.$PATCHLEVEL.$SUBLEVEL-$EXTRAVERSION.

Backup .config

cp .config .config.sav

make mrproper (?)

make dep (not required for 2.6.x)

make clean

Make bzImage (zImage, zdisk, zlilo)

- bzImage uses a different layout and a different loading algorithm (big zImage not bzip2!)

Modules

make modules

make modules_install

- It will be placed in /lib/modules/KERNEL_VERSION

Create Initial Ramdisk

- If you have built your main boot drivers as modules (e.g., SCSI host adapter, filesystem, RAID drivers) then you will need to create an initial RAMdisk image
- Drivers are needed to load the root filesystem but the filesystem cannot be loaded because the drivers are on the filesystem

```
mkinitrd /boot/initrd-2.6.0.img 2.6.0
```

Copy the Kernel and System.map

$ cp arch/i386/boot/bzImage /boot/bzImage-KERNEL_VERSION

$ cp System.map /boot/System.map-KERNEL_VERSION

$ ln -s /boot/System.map-KERNEL_VERSION /boot/System.map

GrUB Configuration

Note that you do not have to rerun grub after making changes to this file

#boot=/dev/hda

default=0

timeout=10

title Red Hat Linux (2.4.20-24.9)

root (hd0,1)

kernel /boot/vmlinuz-2.4.20-24.9 ro root=LABEL=/

initrd /boot/initrd-2.4.20-24.9.img

GrUB Configuration

- Edit the file to include your new kernel information. Keep in mind that GrUB counts starting from 0, so (hd0,1) references the first controller, second partition. If you have created an initial RAMdisk be sure to include it here too. A typical configuration may look something like this:

```
title Test Kernel (2.6.0)
root (hd0,1)
kernel /boot/bzImage-2.6.0 ro root=LABEL=/
initrd /boot/initrd-2.6.0.img
```

RESOURCES

"One of the problems for people wanting to get to know the kernel internals better has been the lack of documentation, and fledging kernel hackers have had to resort to reading the actual source code of the system for most of the details. While I think that is still a good idea,.."

Linus Torvalds

LINUX KERNEL INTERNALS

SECOND EDITION

M BECK, H BOME, M DZIADZKA,
U KUNITZ, R MAGNUS, D VERWORNER

ADDISON-WESLEY

=> www.linuxdoc.org (Kernel Hacker's Guide)

=> kernelnewbie.org (Newbie's to Kernal)

=> Kernel Source and associated documentation in DocBook format

($KERNELDIR)/Documentation

($KERNELDIR)/Docbook

BOOKS

=> Linux Device Drivers

=> Linux Kernel Internals