

Linux Kernel Porting Overview

PART 1 - Boston Linux Users Group, November 2006

Jon Masters <jcm@redhat.com>
<http://people.redhat.com/jcm/>

Obligatory biography

- ◆ Long time enthusiast
- ◆ Embedded Linux
- ◆ Enterprise Linux
- ◆ Articles, books, etc.
- ◆ Works on kernel tools at Red Hat



redhat.com

Overview

- Linux system architecture
- Introducing the Linux kernel
- What is kernel portability?
- Porting the Linux kernel
- Building a complete system
- Q&A

Linux system architecture



redhat.com

Linux System Architecture

Linux system

applications

udev
system utilities

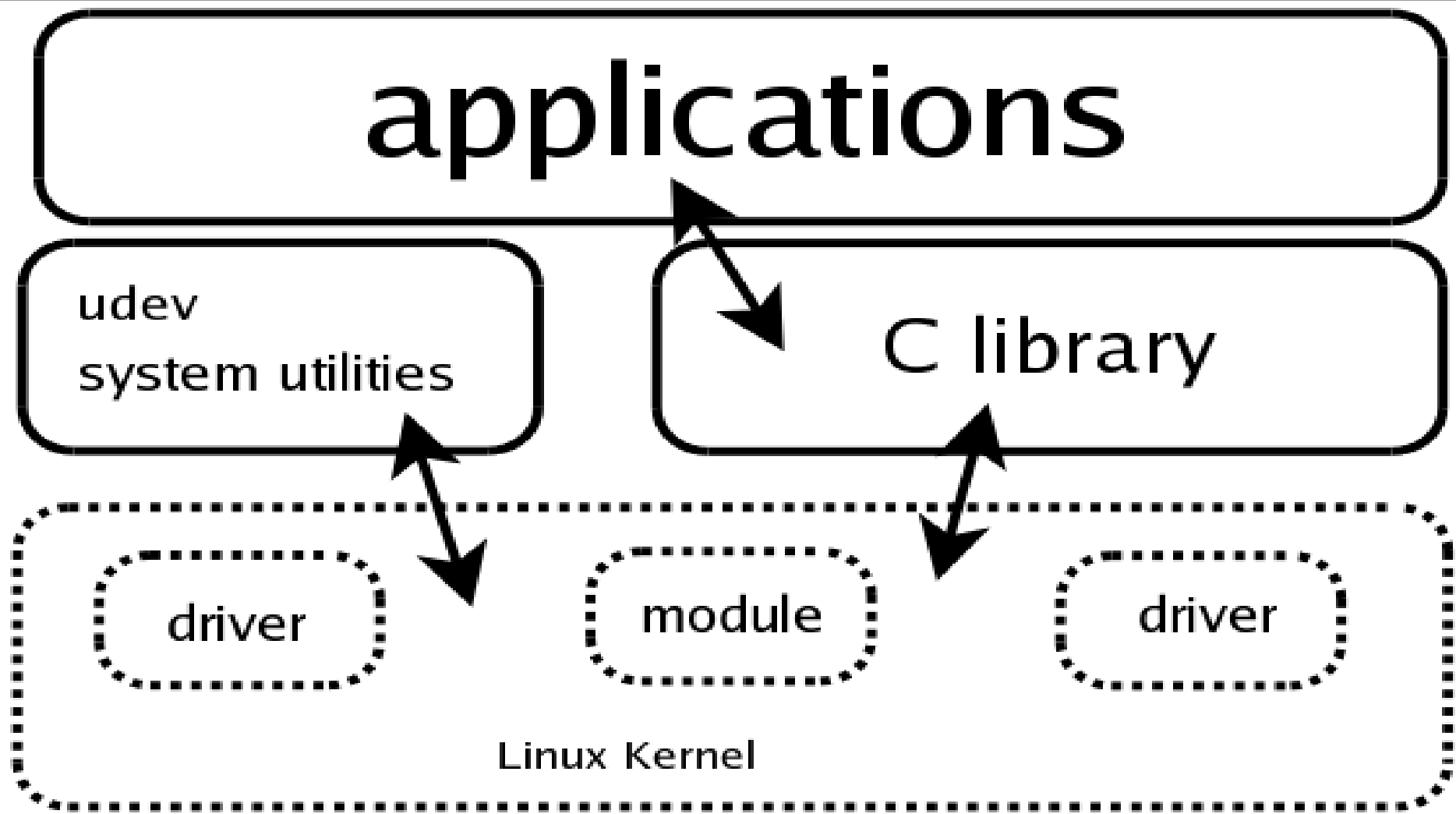
C library

driver

module

driver

Linux Kernel



Applications

- Applications include GNOME Desktop
- Make use of system C library functions
- Library functions are usually wrappers
- Library abstracts user-kernel interface
- User-kernel interface remains stable

System call demo

```
/*
 * syscall demo
 */

#include <stdio.h>
#include <linux/unistd.h>
#include <sys/types.h>

_syscall0(uid_t, getuid);

int main(int argc, char **argv) {
    printf("user ID: %d\n", getuid());
    return 0;
}
```

System call demo

You can build the example as follows:

```
[jcm@maple ~]$ gcc -Wall -o syscall syscall.c  
[jcm@maple ~]$ ./syscall  
user ID: 500
```

The current user ID (500) is displayed.



redhat.com

Linux System Architecture

Linux system

applications

udev
system utilities

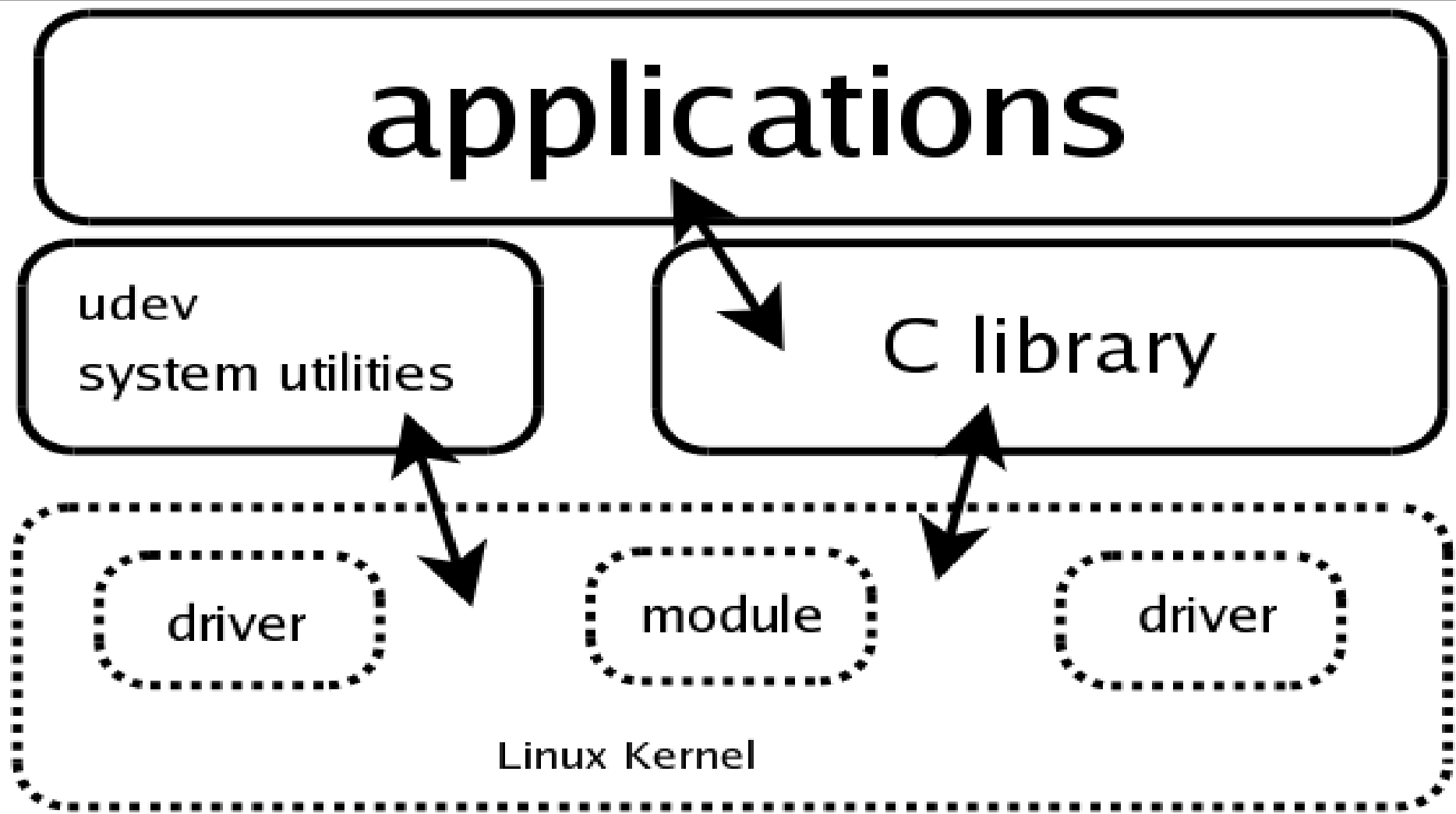
C library

driver

module

driver

Linux Kernel



Linux System Architecture

Linux system

applications

udev
system utilities

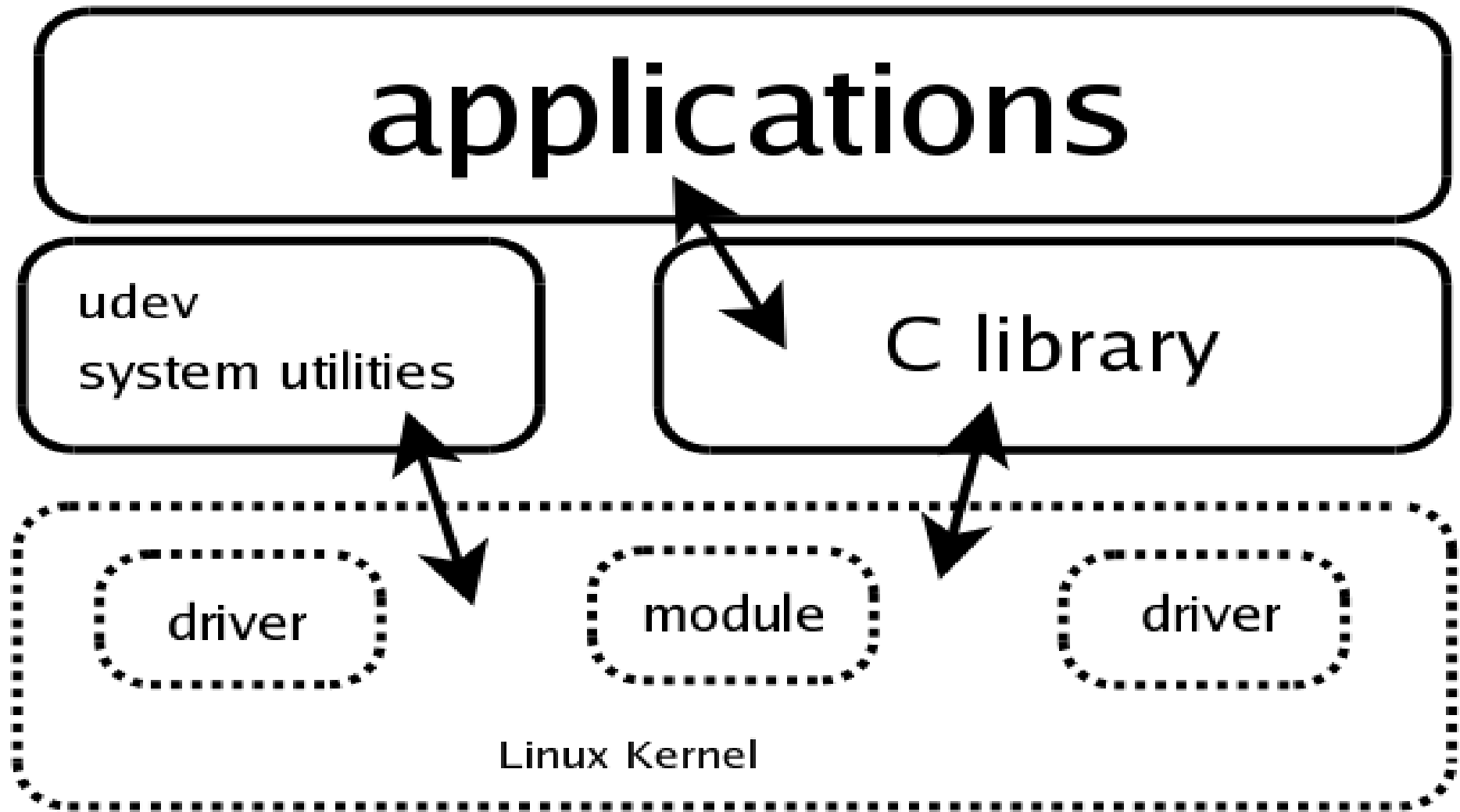
C library

driver

module

driver

Linux Kernel



System tools and utilities

- Kernel provides enhanced functionality
- Events allow for user space notification
- `ioctl()`s enable control over special devices
- Many other tools using `sysfs`, etc.

USB Example

- Insert USB stick
- Kernel driver detects new device
- Kernel notifies udev of new device
- udev loads USB storage driver
- udev notifies the GNOME hald
- hald notifies gvm (over D-BUS)

Linux System Architecture

Linux system

applications

udev
system utilities

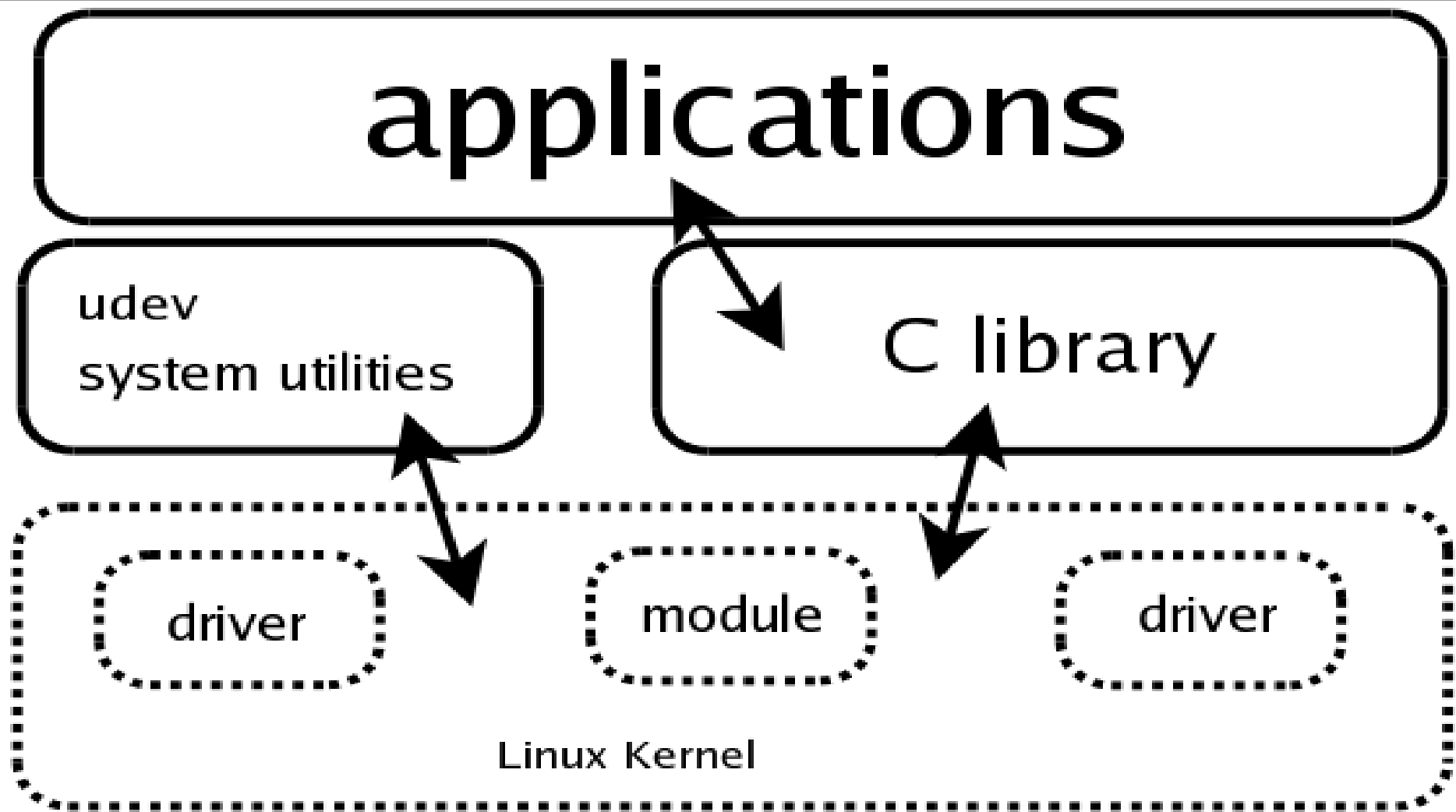
C library

driver

module

driver

Linux Kernel



Introducing the Linux kernel



redhat.com

A brief history of the Linux kernel

- First written for Linus' 80386 in 1991
 - M68K, SPARC, Tivo, Mobile Phones...
- Now supports 20 architectures in upstream
 - Relatively easy to add more
- Kernel module support added early on
 - Use insmod/modprobe at runtime
- Evolving documentation, news sites...

Getting Linux kernel sources

- <http://www.kernel.org/>
- Two ways to get the source
 - Current official stable release (2.6.18.2)
 - Download the tarball/tar.bz2 file and unpack

```
arch      Documentation  ipc        Makefile   REPORTING-BUGS  usr
block    drivers        Kbuild    mm          scripts         vmlinux
COPYING  fs             kernel    Module.symvers security
CREDITS  include       lib       net         sound
crypto   init          MAINTAINERS  README     System.map
```

Tracking development with git

- Track kernel development with git:
 - clone the official Linux kernel tree:
git clone <http://path.to.official.source> linus_26
(see <http://www.kernel.org/git> for sources)
 - Update daily using git pull
 - Follow regular development via lkml.org, LWN and gitk



redhat

Visualizing source changes with gitk

The screenshot shows the gitk application window. The top menu bar includes 'File', 'Edit', and 'Help'. The main window is divided into three panes. The left pane shows a commit history with a vertical axis of colored dots (blue, red, green) and a horizontal axis of commit messages. The middle pane shows a list of commit authors and their email addresses. The right pane shows the commit dates and times. The bottom pane shows the details of the selected commit, including the SHA1 ID, author, committer, parent, child, and merge information. The commit message is also displayed in the bottom pane.

gitk

File Edit Help

[PATCH] Cpuset: fix ABBA deadlock with cpu hotplug li
cpu hotplug: simplify and hopefully fix locking
[cpufreq] ondemand: make shutdown sequence more robust
Merge master.kernel.org:/pub/scm/linux/kernel/git/davem/net-2.6

[TIPC]: Removing useless casts
[IPV4]: Fix nexthop realm dumping for multipath routes
[DUMMY]: Avoid an oops when dummy_init_one() failed
[IFB] After ifb_init_one() failed, i is increased. Decrease
[NET]: Fix reversed error test in netif_tx_trylock
[MAINTAINERS]: Mark LAPB as Oprhan.
[NET]: Conversions from kmalloc+memset to k(z)c)alloc.
[NET]: sun happymeal, little pci cleanup
[IrDA]: Use alloc_skb() in IrDA TX path
[I/OAT]: Remove pci_module_init() from Intel I/OAT DMA engine
[I/OAT]: net/core/user_dma.c should #include <net/netdma.h>
[SCTP]: ADDIP: Don't use an address as source until it is ASCONF-ACK
[SCTP]: Set chunk->data_accepted only if we are going to accept it.
[SCTP]: Verify all the paths to a peer via heartbeat before using them.
[SCTP]: Unhash the endpoint in sctp_endpoint_free().
[SCTP]: Check for NULL arg to sctp_bucket_destroy().
[PKT_SCHED] netem: Fix slab corruption with netem (2nd try)
[WAN]: Converted synclink drivers to use netif_carrier_*()
[WAN]: Cosmetic changes to N2 and C101 drivers
[WAN]: Added missing netif_dormant_off() to generic HDLC
[IPV4]: Get rid of redundant IPCB->opts initialisation

Paul Jackson <pj@sgi.com>
Linus Torvalds <torvalds@macmini.osdl.org>
Linus Torvalds <torvalds@macmini.osdl.org>
Linus Torvalds <torvalds@g5.osdl.org>
Panagiotis Issaris <takis@issaris.org>
Patrick McHardy <kaber@trash.net>
Nicolas Dichtel <nicolas.dichtel@6wind.com>
Nicolas Dichtel <nicolas.dichtel@6wind.com>
Herbert Xu <herbert@gondor.apana.org.au>
David S. Miller <davem@davemloft.net>
Panagiotis Issaris <takis@issaris.org>
Jiri Slaby <jirislaby@gmail.com>
Samuel Ortiz <samuel@sortiz.org>
Henrik Kretzschmar <henne@nachtwindheim.de>
Adrian Bunk <bunk@stusta.de>
Sridhar Samudrala <sri@us.ibm.com>
Sridhar Samudrala <sri@us.ibm.com>
Sridhar Samudrala <sri@us.ibm.com>
Vlad Yasevich <vladislav.yasevich@hp.com>
Sridhar Samudrala <sri@us.ibm.com>
Guillaume Chazarain <guichaz@yahoo.fr>
Krzysztof Halasa <khc@pm.waw.pl>
Krzysztof Halasa <khc@pm.waw.pl>
Krzysztof Halasa <khc@pm.waw.pl>
Herbert Xu <herbert@gondor.apana.org.au>

2006-07-23 14:36:08
2006-07-23 15:12:16
2006-07-23 15:05:00
2006-07-21 19:44:45
2006-07-21 18:52:20
2006-07-21 18:09:55
2006-07-21 18:09:07
2006-07-21 17:56:02
2006-07-21 17:55:38
2006-07-21 17:55:17
2006-07-21 17:51:30
2006-07-21 17:51:02
2006-07-21 17:50:41
2006-07-21 17:50:13
2006-07-21 17:49:49
2006-07-21 17:49:25
2006-07-21 17:49:07
2006-07-21 17:48:50
2006-07-21 17:48:26
2006-07-21 17:45:47
2006-07-21 17:45:25
2006-07-21 17:44:55
2006-07-21 17:41:36
2006-07-21 17:41:01
2006-07-21 17:29:53

SHA1 ID: 12157a8d78af50842774bedb80b7b84a87f60951 Find Exact All fields

Author: Linus Torvalds <torvalds@g5.osdl.org> 2006-07-21 19:44:45
Committer: Linus Torvalds <torvalds@g5.osdl.org> 2006-07-21 19:44:45
Parent: 9df3f3d28bca0157e2bab2f3171d2ad4f0930634 ([TIPC]: Removing useless casts)
Parent: efab4cbe99f9b73d208ad9e5ec9388524005e095 ([SPARC64]: Update defconfig.)
Child: 2cd7cbdf4bd0d0fe58e4dc903e8b413412595504 ([cpufreq] ondemand: make shutdown seq

Merge master.kernel.org:/pub/scm/linux/kernel/git/davem/net-2.6

* master.kernel.org:/pub/scm/linux/kernel/git/davem/net-2.6: (21 commits)
[TIPC]: Removing useless casts
[IPV4]: Fix nexthop realm dumping for multipath routes
[DUMMY]: Avoid an oops when dummy_init_one() failed
[IFB] After ifb_init_one() failed, i is increased. Decrease
[NET]: Fix reversed error test in netif_tx_trylock
[MAINTAINERS]: Mark LAPB as Oprhan.
[NET]: Conversions from kmalloc+memset to k(z)c)alloc.
[NET]: sun happymeal, little pci cleanup
[IrDA]: Use alloc_skb() in IrDA TX path

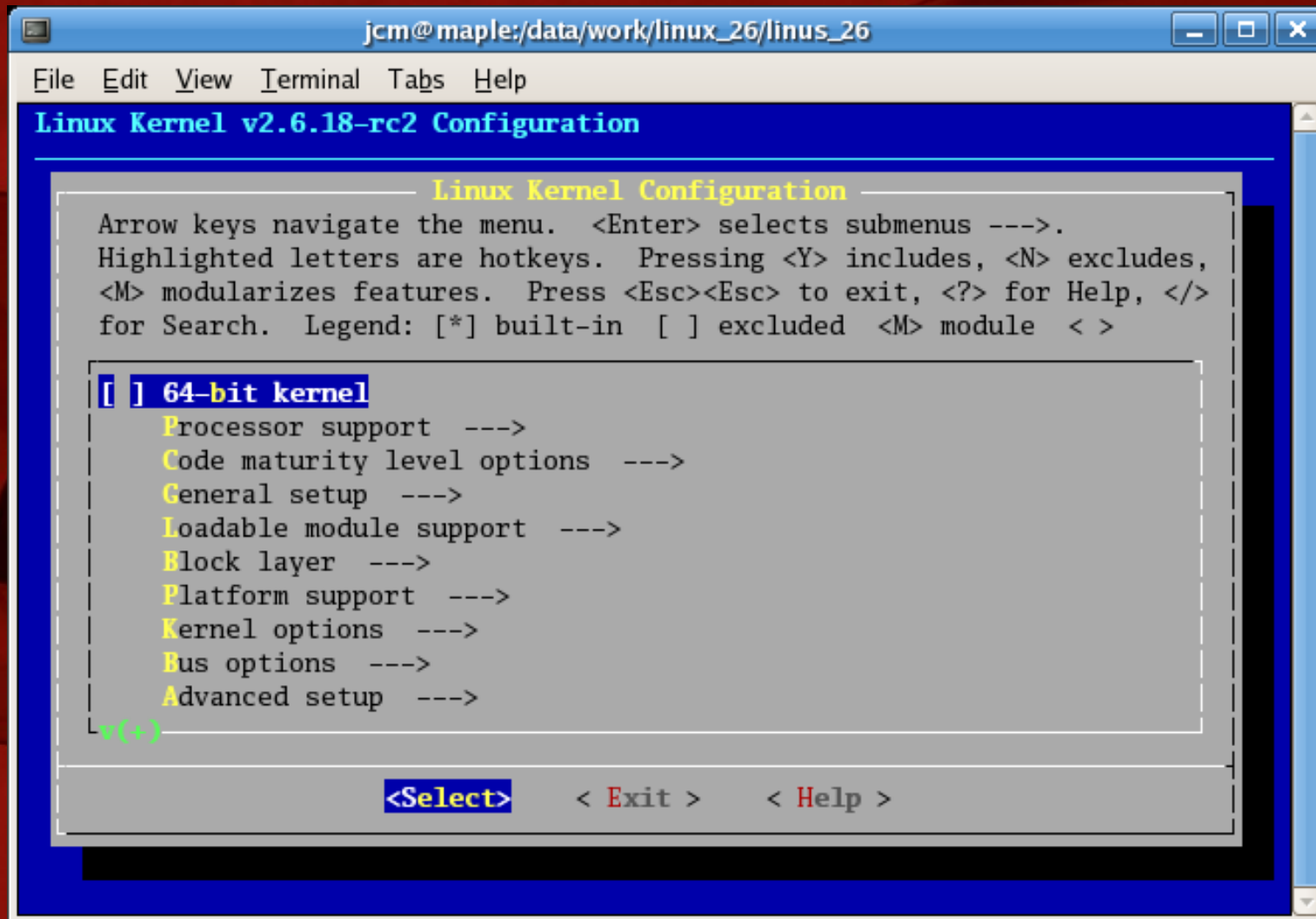
Comments

Vendor sources

- Various patches, not useful for porting to new systems
- Good for writing drivers against specific vendor systems
- Good for building drivers against specific vendor systems
- apt/yum/etc. on a typical desktop system
- For example:
 - `/usr/src/kernels/2.6.18-1.2766.fc6-ppc`

Configuring a Linux kernel

- Configure using make menuconfig:



The screenshot shows a terminal window titled "jcm@maple:/data/work/linux_26/linus_26". The window contains the "Linux Kernel v2.6.18-rc2 Configuration" menu. The menu is displayed in a blue-themed interface with a white background for the main content area. The title "Linux Kernel Configuration" is centered at the top in yellow. Below the title, instructions for navigating the menu are provided: "Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [] excluded <M> module < >". The main menu items are listed in yellow, with "64-bit kernel" highlighted in blue. The items are: "64-bit kernel", "Processor support --->", "Code maturity level options --->", "General setup --->", "Loadable module support --->", "Block layer --->", "Platform support --->", "Kernel options --->", "Bus options --->", and "Advanced setup --->". At the bottom of the menu, there are three options: "<Select>", "< Exit >", and "< Help >".

```
jcm@maple:/data/work/linux_26/linus_26
File Edit View Terminal Tabs Help
Linux Kernel v2.6.18-rc2 Configuration

Linux Kernel Configuration
Arrow keys navigate the menu.  <Enter> selects submenus --->.
Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes,
<M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </>
for Search.  Legend: [*] built-in [ ] excluded <M> module < >

[ ] 64-bit kernel
  Pprocessor support --->
  Ccode maturity level options --->
  Ggeneral setup --->
  Lloadable module support --->
  Bblock layer --->
  Pplatform support --->
  Kkernel options --->
  Bbus options --->
  Aadvanced setup --->
v(+)

<Select>  < Exit >  < Help >
```

Building a Linux kernel

- Build the kernel using make:

```
[jcm@maple linus_26]$ make
```

```
scripts/kconfig/conf -s arch/powerpc/Kconfig
```

```
CHK      include/linux/version.h
```

```
CHK      include/linux/utsrelease.h
```

```
CC       init/main.o
```

```
CHK      include/linux/compile.h
```

```
LD       init/built-in.o
```

```
CC       arch/powerpc/kernel/idle.o
```

```
CC       arch/powerpc/kernel/time.o
```

```
arch/powerpc/kernel/time.c: In function 'get_freq':
```

```
arch/powerpc/kernel/time.c:874: warning: left shift count >= width of  
type
```

```
CC       arch/powerpc/kernel/setup-common.o
```

```
CC       arch/powerpc/kernel/setup_32.o
```

```
...
```

Installing a Linux kernel

- Install modules using `make modules_install`
- Install kernel image in e.g. `/boot`
- Install `System.map` in e.g. `/boot`
- Possibly make an initial ramdisk
- Follow your distribution guide.

Extending the Linux kernel

- Linux supports runtime extension via LKMs.
- Can add new driver/filesystem/etc.
- Cannot change core kernel.
- Sometimes have to modify kernel itself.



redhat.com

Adding your own kernel module

```
/*
 * hello.c - An example kernel module.
 */
#include <linux/module.h>
#include <linux/init.h>

int hello_world(void)
{
    printk("Hello, World!\n");
    return 0;
}

int __init hello_init(void)
{
    printk("Loaded module.\n");
    return 0;
}

void __exit hello_exit(void)
{
    printk("Unloaded module.\n");
}
```

```
/* Module Metadata */

MODULE_AUTHOR("Jon Masters <jcm@redhat.com>");
MODULE_DESCRIPTION("module example");
MODULE_LICENSE("GPL");

module_init(hello_init);
module_exit(hello_exit);

/* Exported Functions */

EXPORT_SYMBOL_GPL(hello_world);
```

Figure: hello.c source code

Building your own kernel module

Add the following to a Kbuild file:

```
obj-m          := hello_module.o
hello_module-y += hello.o
```

Build the module against an existing kernel:

```
[jcm@maple test-1.0]$ make -C /usr/src/kernels/2.6.18-1.2766.fc6-ppc M=$PWD
make: Entering directory `/usr/src/kernels/2.6.18-1.2766.fc6-ppc'
LD /home/jcm/test-1.0/built-in.o
CC [M] /home/jcm/test-1.0/hello.o
LD [M] /home/jcm/test-1.0/hello_module.o
Building modules, stage 2.
MODPOST
CC /home/jcm/test-1.0/hello_module.mod.o
LD [M] /home/jcm/test-1.0/hello_module.ko
make: Leaving directory `/usr/src/kernels/2.6.18-1.2766.fc6-ppc'
```

Installing your own kernel module

Install the module into global `/lib/modules`:

```
[jcm@maple test-1.0]$ sudo make -C /usr/src/kernels/2.6.18-1.2766.fc6-ppc modules_install M=$PWD
Password:
make: Entering directory `/usr/src/kernels/2.6.18-1.2766.fc6-ppc'
INSTALL /home/jcm/test-1.0/hello_module.ko
DEPMOD 2.6.18-1.2766.fc6
make: Leaving directory `/usr/src/kernels/2.6.18-1.2766.fc6-ppc'
```

Rebuild global module dependencies:

```
[jcm@maple test-1.0]$ sudo /sbin/depmod -a
```

Loading and unloading your kernel module

Load the module:

```
[jcm@maple test-1.0]$ sudo /sbin/modprobe hello_module
```

Check kernel ringbuffer:

```
[jcm@maple test-1.0]$ dmesg|tail  
Loaded module.
```

Unload the module:

```
[jcm@maple test-1.0]$ sudo /sbin/modprobe -r hello_module
```

Linux Kernel Architecture



redhat.com

Linux Kernel Architecture

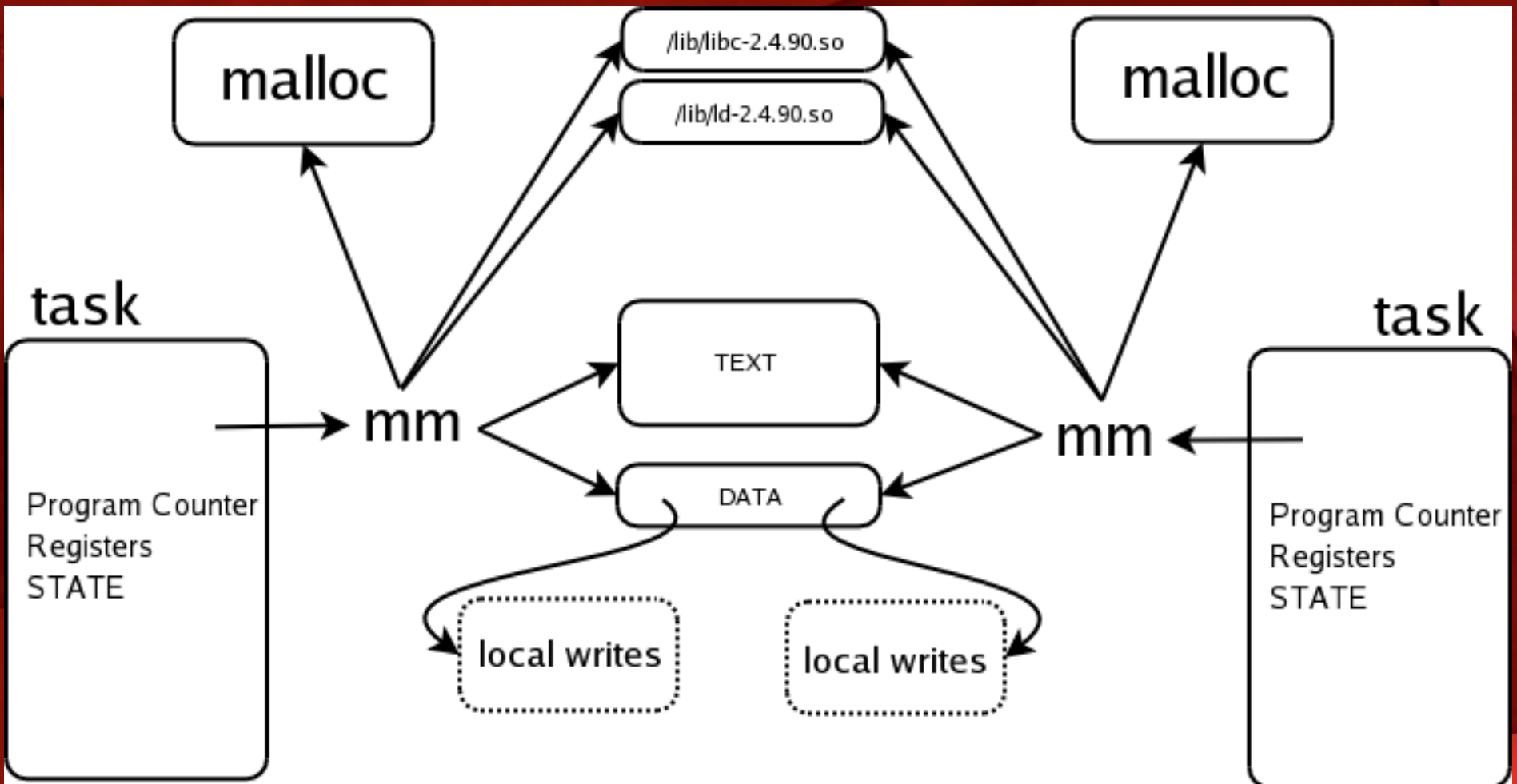
- The kernel as a privileged software library
- Asynchronous entry
 - Interrupts
(timer, hard disk, network...)
- Synchronous entry
 - System calls
(open, read, write, ptrace...)

The kernel as a privileged library

- The kernel is not magical
- Performs tasks on our behalf
- Maintains fundamental software abstraction
- Limits access to shared resources
- Is there to help us, not to hinder us

The process abstraction

- Every running program represented by process.
 - Referred to within the kernel as a task.



Asynchronous Kernel Entry

- Things that happen involving hardware
- Hardware defined trap in response:

```
/* 0x1000 - Programmable Interval Timer (PIT) Exception */  
START_EXCEPTION(0x1000, Decrementer)  
NORMAL_EXCEPTION_PROLOG  
lis    r0,TSR_PIS@h  
mtspr  SPRN_TSR,r0          /* Clear the PIT exception */  
addi   r3,r1,STACK_FRAME_OVERHEAD  
EXC_XFER_LITE(0x1000, timer_interrupt)
```

Synchronous Kernel Entry

- Things that we caused to happen directly
- System Calls
 - depending upon architecture
- Program Exceptions
- Data Access Exceptions
 - stack growth, paging
- Emulation
 - traps for VMs, FP...

Kernel Sources

arch

crypto

include

kernel

mm

scripts

block

Documentation

init

lib

net

security

COPYING

drivers

ipc

MAINTAINERS

README

sound

CREDITS

fs

Kbuild

Makefile

REPORTING-BUGS

usr

What is kernel portability?



redhat.com

What is portability?

- “Writing software that can be more easily ported from one architecture (or platform) to another without rewriting it from scratch”.
- Abstract away 64/32-bit, endian issues, etc.
- Linux has no official HAL
 - No generic “Linux Driver Model” either.

Architecture Ports

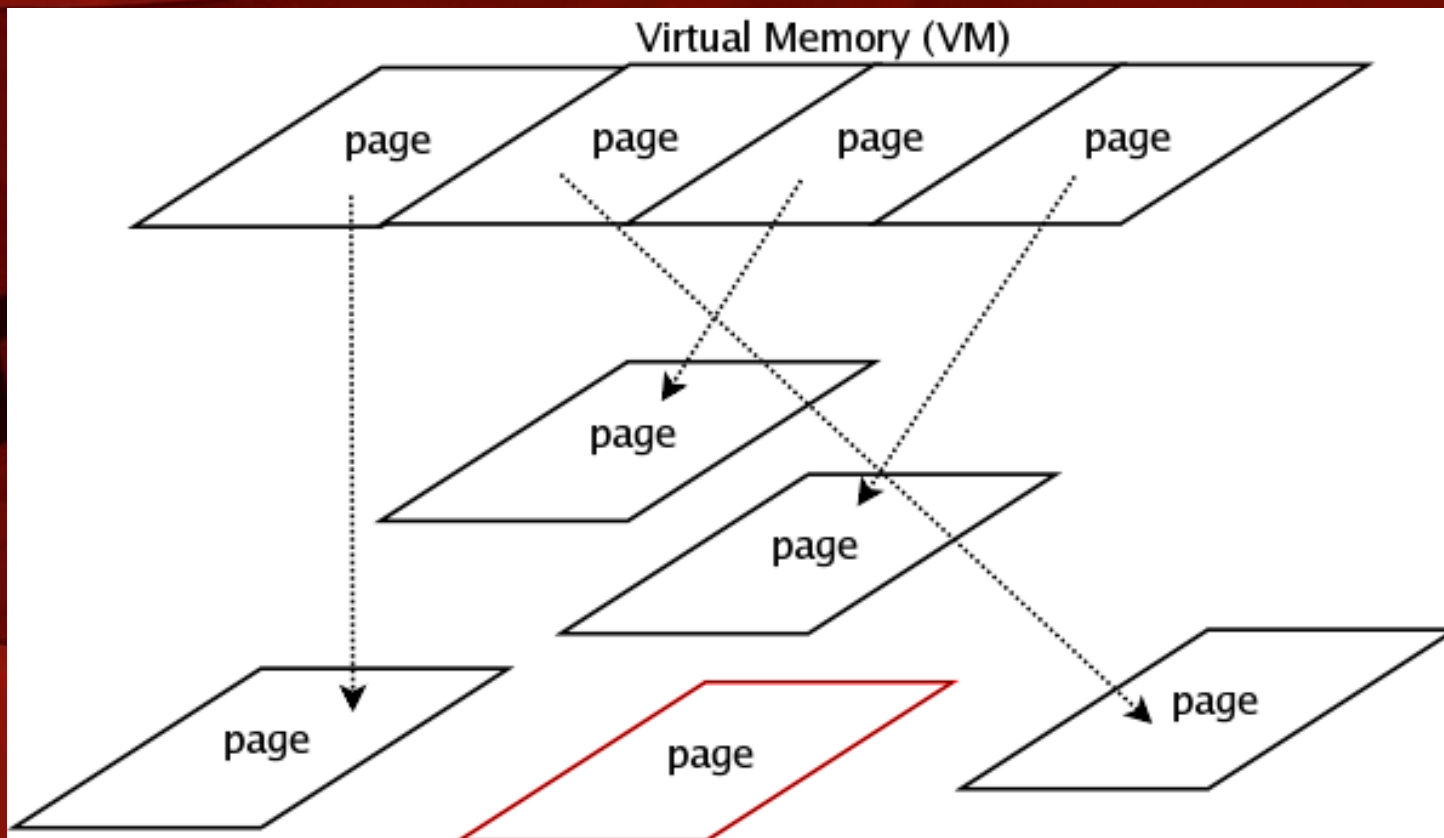
- What's a machine architecture?
- Linux on whole new type of hardware
 - Intel IA32 (“x86”), IA64, POWER, PowerPC, SPARC, etc.
- Typically done by hardware vendors
 - Vested interest in getting support in order to drive sales.
- May require lots of funky hardware
 - Debuggers
- Need to first port GCC, other tools

Examples of architectural differences

- What is virtual memory?
- Why does it differ between architectures?
- How does Linux implement virtual memory?
- Examples

What is virtual memory?

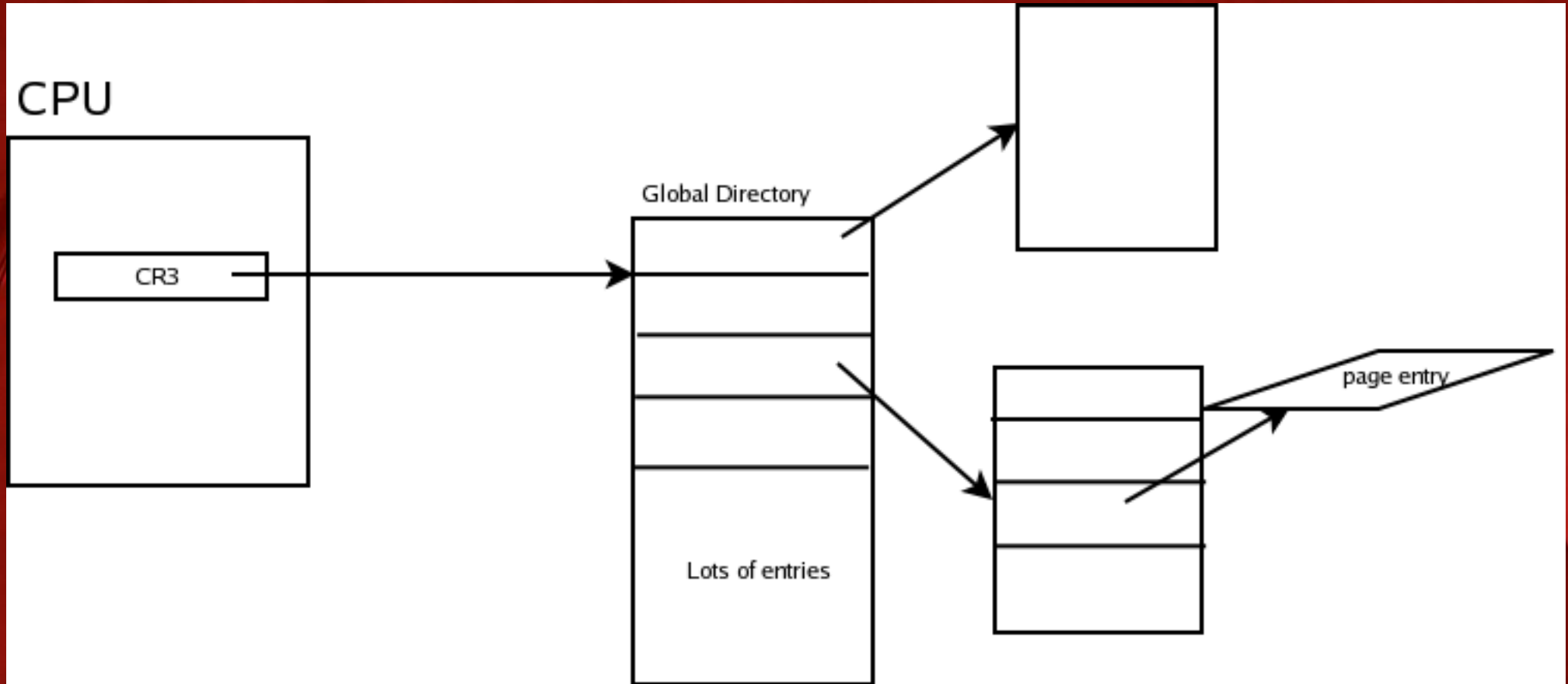
- Abstraction designed to manage memory.
- Group physical memory into pages:



Virtual Memory (VM) architectural differences

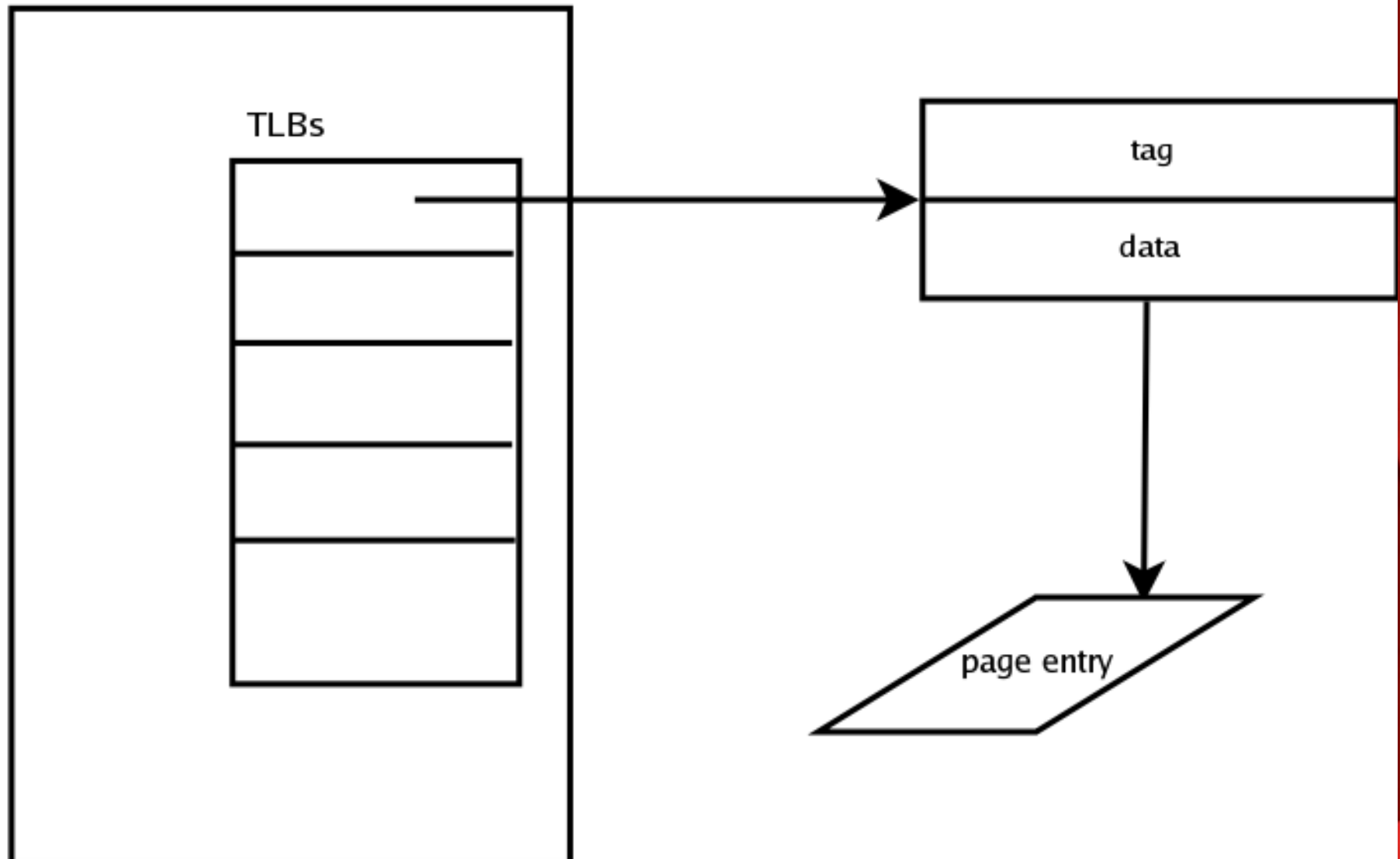
- Virtual memory is just a concept.
- Everyone implements it differently.
- Intel giant page table directory.
- PowerPC giant hash table.
- PowerPC self-managed TLBs.

Intel virtual memory...



Embedded PowerPC virtual memory...

Embedded PowerPC (ppc4xx)



Tools needed to build the kernel

- Need a compiler to build a kernel.
- Binutils provides many ELF binary tools
- Check out dummy build/link build hacks
- Need a lot of documentation/expertise
- Try running objdump on your kernel.

objdump -x -D -S vmlinux...

```
/data/work/linux_26/linus_26/vmlinux: file format elf32-powerpc  
/data/work/linux_26/linus_26/vmlinux  
architecture: powerpc:common, flags 0x00000112:  
EXEC_P, HAS_SYMS, D_PAGED  
start address 0xc0000000
```

Program Header:

```
LOAD off 0x00010000 vaddr 0xc0000000 paddr 0xc0000000 align 2**16  
filesz 0x003b0000 memsz 0x003ffcb8 flags rwx  
STACK off 0x00000000 vaddr 0x00000000 paddr 0x00000000 align 2**2  
filesz 0x00000000 memsz 0x00000000 flags rwx
```

Sections:

Idx	Name	Size	VMA	LMA	File off	Algn
0	.text	002a3000	c0000000	c0000000	00010000	2**5
			CONTENTS, ALLOC, LOAD, READONLY, CODE			
1	.rodata	00057580	c02a3000	c02a3000	002b3000	2**3
			CONTENTS, ALLOC, LOAD, READONLY, DATA			
2	.pci_fixup	00000470	c02fa580	c02fa580	0030a580	2**2
			CONTENTS, ALLOC, LOAD, READONLY, DATA			
3	__ksymtab	00004c18	c02fa9f0	c02fa9f0	0030a9f0	2**2
			CONTENTS, ALLOC, LOAD, DATA			

objdump -x -D -S vmlinux...

c0000000 <_start>:

```
c0000000: 60 00 00 00  nop
c0000004: 60 00 00 00  nop
c0000008: 60 00 00 00  nop
```

c000000c <__start>:

```
c000000c: 2c 05 00 00  cmpwi  r5,0
c0000010: 41 82 00 0c  beq-   c000001c <__start+0x10>
c0000014: 48 32 51 05  bl     c0325118 <prom_init>
c0000018: 7f e0 00 08  trap
c000001c: 3f e0 42 6f  lis    r31,17007
c0000020: 63 ff 6f 58  ori    r31,r31,28504
c0000024: 7c 03 f8 00  cmpw   r3,r31
c0000028: 40 82 00 0c  bne-   c0000034 <__start+0x28>
c000002c: 48 32 e0 09  bl     c032e034 <bootx_init>
c0000030: 7f e0 00 08  trap
c0000034: 7c 7f 1b 78  mr     r31,r3
c0000038: 7c 9e 23 78  mr     r30,r4
c000003c: 3b 00 00 00  li    r24,0
c0000040: 48 32 3e 79  bl     c0323eb8 <early_init>
c0000044: 48 00 39 21  bl     c0003964 <mmu_off>
```

objdump -x -D -S vmlinux...

c0003800 <start_here>:

```
c0003800: 3c 40 c0 34  lis  r2,-16332
c0003804: 60 42 e0 50  ori  r2,r2,57424
c0003808: 3c 82 40 00  addis r4,r2,16384
c000380c: 38 84 01 d0  addi  r4,r4,464
c0003810: 7c 93 43 a6  mtsprg 3,r4
c0003814: 38 60 00 00  li   r3,0
c0003818: 7c 72 43 a6  mtsprg 2,r3
c000381c: 3c 20 c0 3a  lis  r1,-16326
c0003820: 38 21 00 00  addi  r1,r1,0
c0003824: 38 00 00 00  li   r0,0
c0003828: 94 01 1f f0  stwu  r0,8176(r1)
c000382c: 7f e3 fb 78  mr   r3,r31
c0003830: 7f c4 f3 78  mr   r4,r30
c0003834: 48 32 06 ed  bl   c0323f20 <machine_init>
c0003838: 48 00 a3 2d  bl   c000db64 <__save_cpu_setup>
c000383c: 48 32 54 d1  bl   c0328d0c <MMU_init>
c0003840: 3c 80 c0 00  lis  r4,-16384
c0003844: 60 84 38 5c  ori  r4,r4,14428
c0003848: 3c 84 40 00  addis r4,r4,16384
c000384c: 38 60 10 02  li   r3,4098
c0003850: 7c 9a 03 a6  mtsrr0 r4
```

Architecture Ports

- Machine defined hardware entry
 - Linux kernel provides `head.S`, `entry.S`, etc.
 - See `arch/$my_arch/kernel`
- Machine defined interrupts and exceptions
 - What's an exception?
- Machine defined Virtual Memory
 - Huh?
- End up at `start_kernel` eventually

Platform Ports

- What's a platform anyway?
- Based on standard architecture
 - Much more common than arch. ports
(run Linux on your own hardware based on COTS parts)
- Use application-dependent devices
- Different memory layout
- Huh?

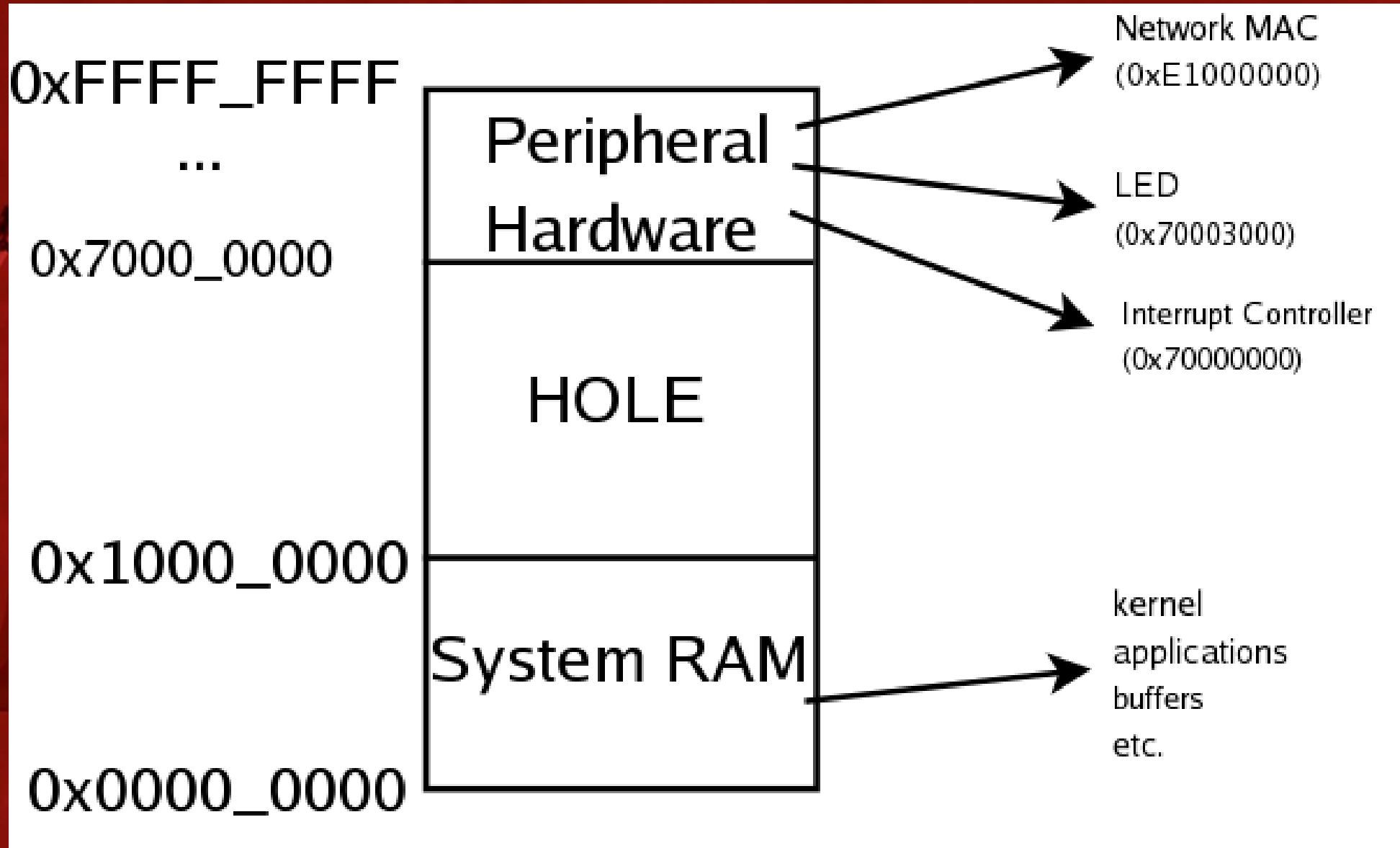
Platform Example - PowerPC

- Tivo PVR runs Linux on PowerPC platform
- Powerbooks Linux on a PowerPC platform
- Industrial Linux on a PowerPC platform
- GameCube Linux on a PowerPC platform
- All Based on a standard architecture

Platform Example - ARM

- PDA running Linux on an ARM platform
- iPod running Linux on an ARM platform
- Broadcom WiFi on an ARM platform
- Your cell phone (running Linux yet?)
- All Based on a standard architecture

Platform Example



Porting the Linux kernel



redhat.com

Determining the hardware platform

- Select a hardware platform
 - Processor/architecture requirements
 - Memory, storage, other requirements
- Acquire necessary hardware tools
 - Hardware debugger (e.g. BDI2000)



redhat.com

Acquire necessary software tools

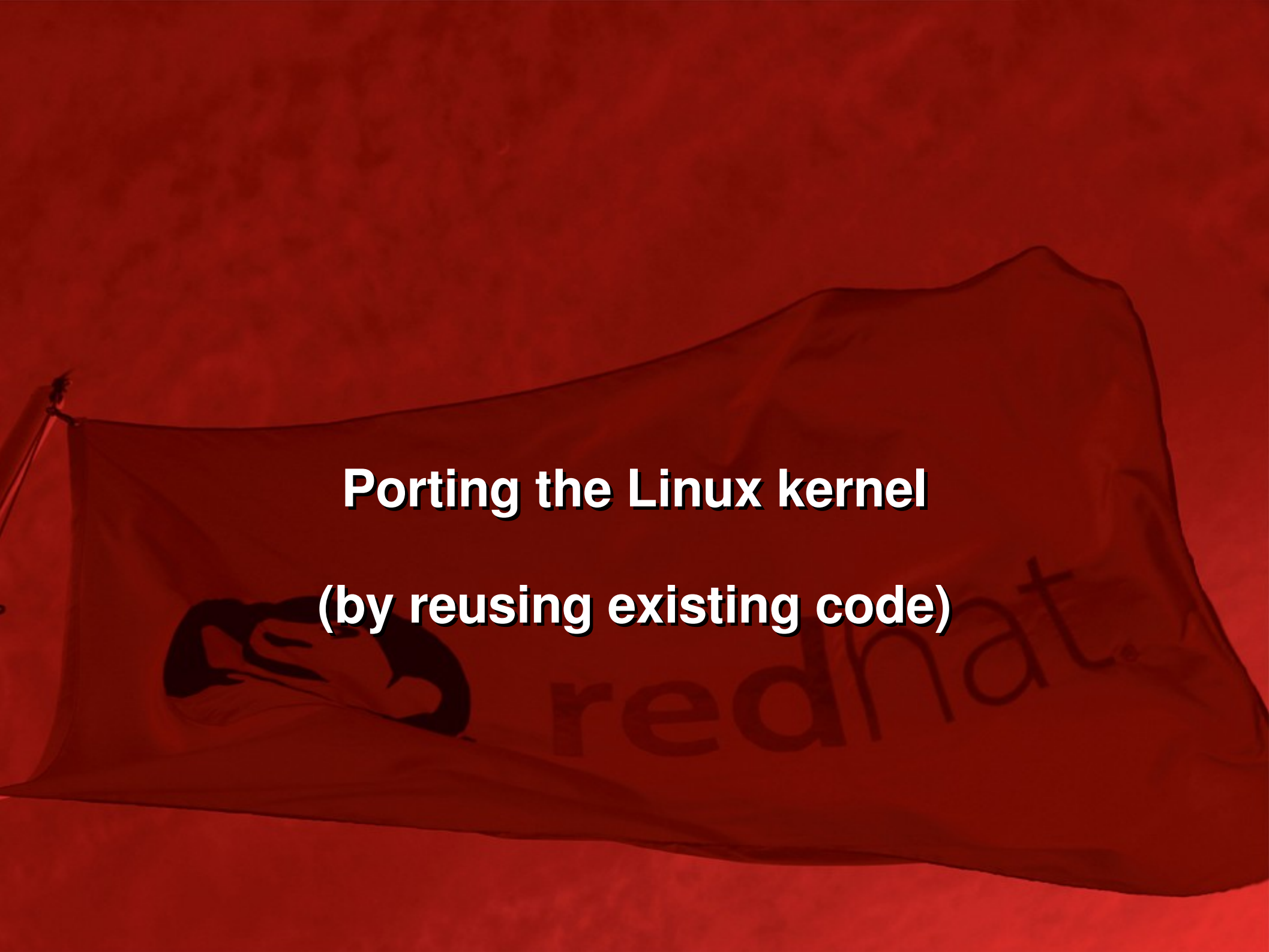
- You'll need a good toolchain
 - for x86 (IA32), grab one from your distro.
 - otherwise, you need to build/find one.
- Dan Keigel's crosstool is the most popular:
 - <http://www.kegel.com/crosstool/>
- Vendors sell tools with proprietary gloss.
 - Can buy Eclipse based tools.

Get a good book

- You need to understand the kernel
 - Understanding the Linux Kernel (O'Reilly)
 - Linux Kernel Development (Novell)
 - Linux Device Drivers (O'Reilly)
- If you're working on embedded devices:
 - Building Embedded Linux Systems

Bringing up the system

- Need a workable firmware/BIOS.
 - e.g. uboot, redboot, LinuxBIOS, etc.
- Port firmware to device if needed.
- Ensure necessary toolchain available.
 - What's a toolchain?
- Ensure debugging tools available.
- Spend a lot of time hacking code.

A red flag with the Linux logo and the word 'redhat' on it, set against a red background. The flag is slightly wrinkled and appears to be waving. The Linux logo is on the left, and the word 'redhat' is on the right. The text 'Porting the Linux kernel (by reusing existing code)' is centered on the flag.

Porting the Linux kernel
(by reusing existing code)

Follow the standard formula

- Most systems are variants of existing ones
- Copy the base platform files and modify
- Add drivers for custom hardware devices



redhat

PowerPC Example

- PowerPC architecture/platform files in:
include/asm-ppc and arch/ppc
(ongoing work to replace ppc with powerpc)
- System boots through to platform_init()
- We supply platform_init and use it to:
 - Educate Linux about hardware
 - Setup the kernel environment
 - Workaround any known issues

Platform Porting – PowerPC Example

```
/* Clock Defines
 */
#define MY_PLATFORM_SYSCLK          (300000000)
/* 300MHz Clock */

/*****

 * MYPLATFORM Physical memory map *

*****/

*/

#define MYPLATFORM_BRAM_BASE_PADDR  0xFFFFC000
#define MYPLATFORM_LED_BASE_PADDR   0x70003000
#define MYPLATFORM_UART0_BASE_PADDR 0x70002000
#define MYPLATFORM_BUTTON_BASE_PADDR 0x70001000
#define MYPLATFORM_XINTC0_BASE_PADDR 0x70000000
```

Platform Porting – PowerPC Example

- arch/ppc/platforms/my_platform.c:
- Void __init platform_init()
 - Called by the kernel early on to setup low-level hardware
 - Registers contain pointers to board info (binfo), initrd, etc.
- Save board information from bootloader
- Setup initrd
- Produce some debugging output (no printk)

Platform Porting – PowerPC Example

- `platform_init` also sets up board functions:
 - `ppc_md.setup_arch`
 - `ppc_md.show_percpuinfo`
 - `ppc_md.init_IRQ`
 - `ppc_md.get_irq`
 - `ppc_md.restart`
 - `ppc_md.time_init`
 - `ppc_md.find_end_of_memory`
 - `ppc_md.setuo_io_mappings`
- Kernel will use these functions as callbacks later

Platform Porting – Device Drivers

- Most platforms are very similar
 - Different memory layout, devices, etc.
- Drivers often very different
 - proprietary?!!!
- Use generic PCI/USB/other subsystems
- Add support for device to existing driver
- User-space device drivers?

Building a complete system



redhat.com

Integration

- System is more than a kernel
- Need a working userland (distribution)
- Need to handle software updates



Need a userland

- System is useless without init/bash/something.
 - did you want a webserver with that?
- Can build your own distribution from scratch.
- Can use an existing vendor distribution.
- Can use PTXdist/busybox as a base.
- Experiment!

Handling Updates

- Probably should think about updates
- Build your system with a web server
 - everyone else does.
- Design platform for durability
 - keep a “known good” configuration.
 - keep a spare flash image/kernel.

Jon's Personal Experiences

- Linux in NMR
- Supporting Embedded Linux
- Hobbyist Homebrew



Q&A

- See also:
 - <http://www.lwn.net/>
 - <http://www.lkml.org/>
- Mail me with stuff we can't cover here.
- Obligatory legal disclaimer.