Linux Kernel Modules
&
Device Drivers

April 9, 2012

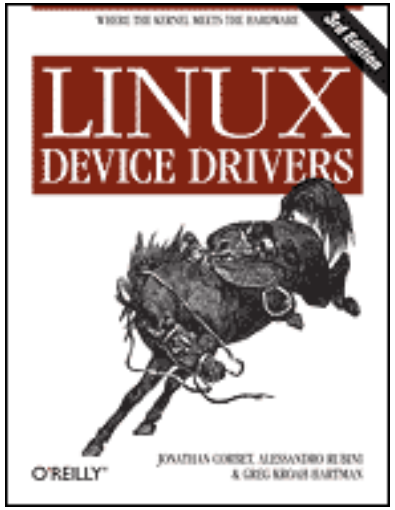
Resources

- Linux Device Drivers,3rd Edition, Corbet, Rubini, Kroah-Hartman; O'Reilly

---

  - kernel 2.6.10
    - we will use 3.1.9

The current kernel APIs are different.



  - http://lwn.net/Kernel/LDD3/
  - Creative Commons Attribution-ShareAlike 2.0 license
  - Chapters **1, 2**, 3, 5, 18

Linker & Libraries Guide (Everything you ever wanted to know about ELF but never asked)
http://docs.oracle.com/cd/E19963-01/html/819-0690/index.html
http://docs.oracle.com/cd/E19963-01/pdf/819-0690.pdf

http://kernelnewbies.org/KernelGlossary

http://kernel.org/

http://tldp.org/LDP/lkmpg/2.6/html/lkmpg.html

Linux

- Loadable Modules

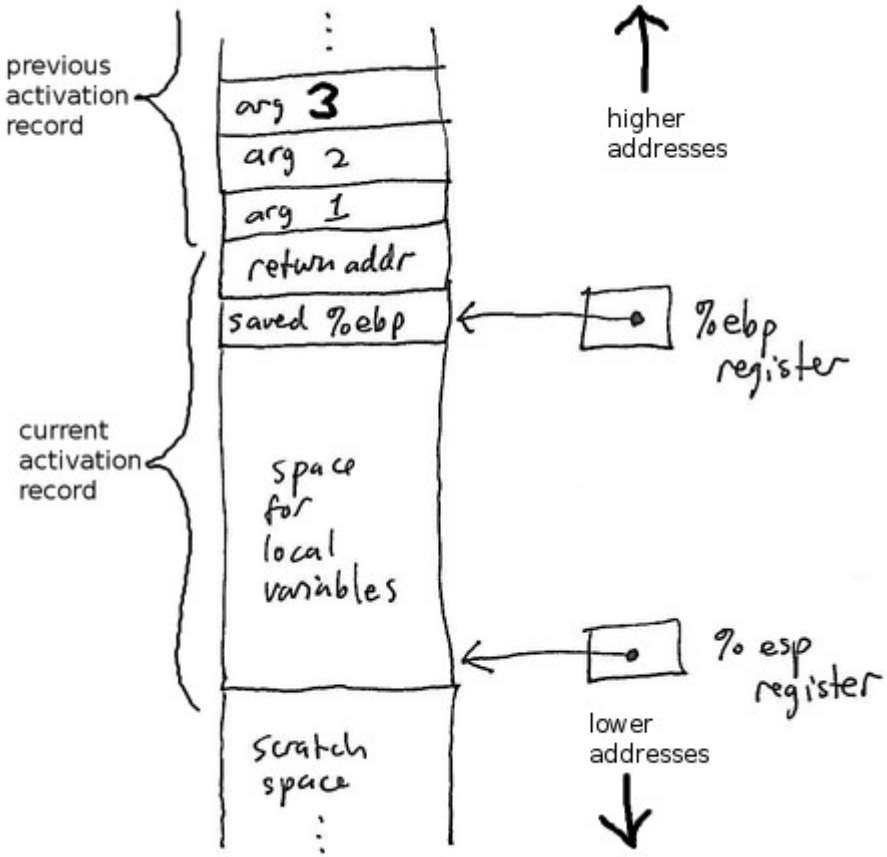
- Device Drivers
  - character devices
  - block devices
  - network devices

Looks like a file. You supply the implementation for each of the file operations that could be called on the device.

Security

- Device Drivers run in the Kernel!

- Buffer Overflow



http://faculty.ycp.edu/~dhovemey/fall2009/cs340/lecture/lecture23.html

User Space Device Driver

- http://www.kernel.org/doc/htmldocs/uio-howto.html

Implementation

- In the Kernel
  - no libc

- Kernel Headers

Structure

- Init/exit functions (*loadable* module)

Register

- License

Example

http://lwn.net/images/pdf/LDD3/ch02.pdf

http://lxr.linux.no/linux+v3.1.9/include/linux/module.h#L114
http://lxr.linux.no/linux+v3.1.9/include/linux/printk.h

```
#include <linux/init.h>
#include <linux/module.h>

MODULE_LICENSE("Dual BSD/GPL");

static int __init  hello_init(void)
{
    printk(KERN_ALERT "Hello, world\n");
    return 0;
}

static void __exit hello_exit(void)
{
    printk(KERN_ALERT "Goodbye, cruel world\n");
}

module_init(hello_init);
module_exit(hello_exit);
```

Makefile

http://www.gnu.org/software/make/manual/make.html#toc_Running

```
obj-m += kbleds.o

all:
    make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules

clean:
    make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean
```

-C change to directory before doing anything else
M= creates a variable for use in the Makefile

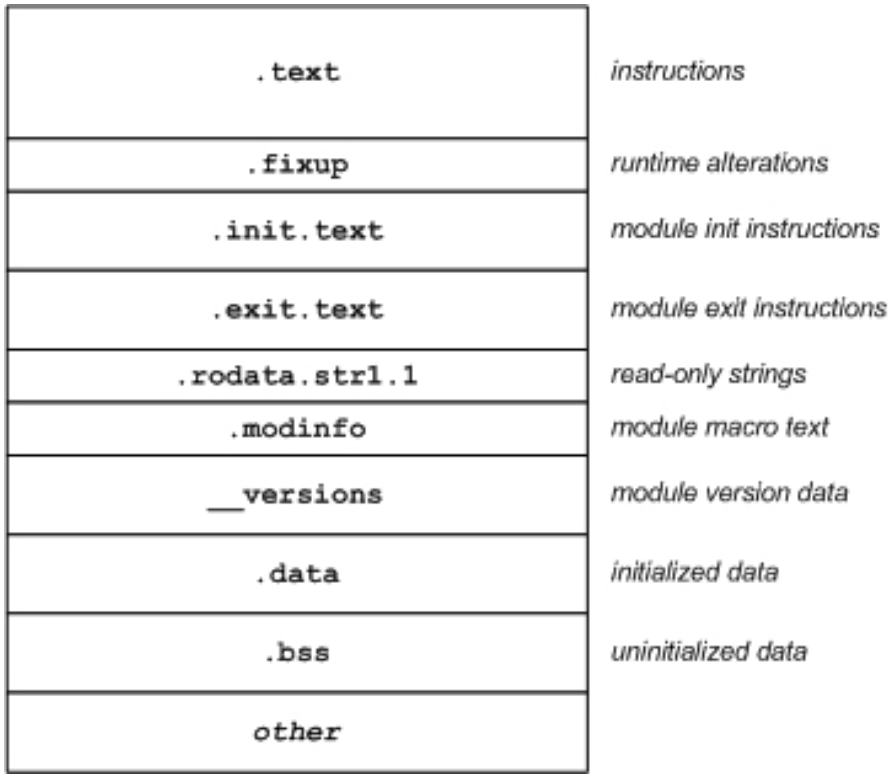
```
make -C /lib/modules/3.1.9-2-ARCH/build M=/home/cs460/kbleds modules
```

```
ifeq ("$(origin M)", "command line")
  KBUILD_EXTMOD := $(M)
endif
```

http://www.gnu.org/software/make/manual/make.html#Origin-Function

ELF!

module.ko



http://www.ibm.com/developerworks/linux/library/l-lkm/

Concurrency

- Must be threadsafe

- Functionality available as soon as registered
  - races in init
  - errors in init

Linux Commands

- insmod
- modprobe
- rmmod
- lsmod
- modinfo

Loading a Module

http://www.ibm.com/developerworks/linux/library/l-lkm/
http://lxr.linux.no/linux+v3.1.9/kernel/module.c#L2803
http://lxr.linux.no/linux+v3.1.9/kernel/module.c#L2949

---

sys_init_module(mod_name, args)
1) /* Permissions Checks */
2) mod = load_module (mod_name, args)
3) /* Add module to linked list */
4) /* Call module notify list with state change */
5) /* call the module's init function */
   mod−>init()
6) mod−>state = MODULE_STATE_LIVE
7) return

load_module(mod_name, args)
1) Allocate temp memory, read ELF
2) Sanity checks (bad object, arch,)
3) Map ELF section headers to variables
4) Read in optional module args
5) mod->state = MODULE_STATE_COMING
6) Allocate Per CPU sections
7) Allocate final module memory
8) Move SHF_ALLOC sections from temp to final module memory
9) Fixup symbols and perform relocations
10) Flush instruction cache
11) Clean up (free temp memory) and return module

Character Devices

- Read/write character (byte) streams

---

ls -al /dev
Look for c (character) or b (block)
Major/minor number

- cat /proc/devices

- mknod /dev/X {c,b} Major# Minor#

file_operations

- linux/fs.h
- http://lxr.linux.no/linux+v3.1.9/include/linux/fs.h#L1563

- Function pointers to implementation for this Device Driver
  open
  release
  read
  write

  lseek

  mmap

  flush
  and more.....

ioctl

http://lxr.linux.no/linux+v3.1.9/include/asm-generic/ioctl.h

- Input/output control
- Device specific system call

- man ioctl

```
// POSIX
#include <stropts.h>

int ioctl(int fildes, int request, ... /* arg */);
```

```
#include <sys/ioctl.h>

int ioctl(int d, int request, ...);
```

struct file

- linux/fs.h
- *open* file

- http://lxr.linux.no/linux+v3.1.9/include/linux/fs.h#L953

struct inode

```
ls -i
stat data.txt
```

- linux/fs.h
  http://lxr.linux.no/linux+v3.1.9/include/linux/fs.h#L748

- a file
  - many struct file per one inode

http://www.ibm.com/developerworks/aix/library/au-speakingunix14/

Kernel

- Register your Device driver with the kernel

Concurrency Primatives

- mutex/semphore (up, down_interruptable)

struct timer_list

- http://lxr.linux.no/linux+v3.1.9/include/linux/timer.h#L12

---

init_timer

add_timer

del_timer

http://lwn.net/images/pdf/LDD3/ch07.pdf

struct tty_driver

- http://lxr.linux.no/linux+v3.1.9/include/linux/tty_driver.h
  http://lxr.linux.no/linux+v3.1.9/drivers/tty/vt/vt_ioctl.c

- tty: teletype
- connected to a console
  - now a virtual console (vc)

http://lwn.net/images/pdf/LDD3/ch18.pdf

http://www.linusakesson.net/programming/tty/index.php

Wednesday

http://archlinux.org/

https://wiki.archlinux.org/index.php/Main_Page

- Run ArchLinux inside of VirtualBox

- You will have root access
- We will reuse this VirtualBox machine for a series of labs
- ArchLinux: a little more hacker-ish than OpenSUSE
  - great for learning all the behinds the scene internals